



FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING



Master's Thesis

Volume invariant Cosserat rods for position-based (PBD) muscle deformation

Duc Long Hoang





**FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA**

**DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING**

Master's Thesis

Volume invariant Cosserat rods for position-based (PBD) muscle deformation

Bc. Duc Long Hoang

Thesis advisor

doc. Ing. Josef Kohout, Ph.D.

© 2026 Duc Long Hoang.

All rights reserved. No part of this document may be reproduced or transmitted in any form by any means, electronic or mechanical including photocopying, recording or by any information storage and retrieval system, without permission from the copyright holder(s) in writing.

Citation in the bibliography/reference list:

HOANG, Duc Long. *Volume invariant Cosserat rods for position-based (PBD) muscle deformation*. Pilsen, Czech Republic, 2026. Master's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor doc. Ing. Josef Kohout, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2025/2026

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Bc. Duc Long HOANG
Osobní číslo:	A23N0017P
Studijní program:	N0613A140040 Softwarové a informační systémy
Téma práce:	Objemově nezávislé Cosseratovo tyče pro position-based (PBD) deformaci svalů
Zadávací katedra:	Katedra informatiky a výpočetní techniky

Zásady pro vypracování

1. Seznamte se s projektem MuscleWrapping 2.0 vyvíjeném na Katedře informatiky a výpočetní techniky.
2. Popište dosavadní přístupy k modelování svalů, přičemž se zaměřte na přístupy založené na metodě PBD nebo využívajících Cosseratovo tyče.
3. Navrhněte novou metodu pro reprezentaci svalů množinou objemově nezávislých neprotínajících se Cosseratovo tyčí, které budou využívat znalosti o úrovni aktivace svalu pro realističtější změnu svého tvaru.
4. Rozšiřte metodu o zajištění, že se tyče v průběhu simulace nebudou vzájemně protínat, ani nebudou protínat okolní tkáň (zejména kosti).
5. Provedte implementaci navržené metody v prostředí MuscleWrapping 2.0, s využitím již dostupného kódu. Dbejte na podporu budoucího využívání a úprav implementace.
6. Vytvořené programové řešení ověřte na uměle vytvořených scénářích a následně otestujte pro různé pohyby dolní končetiny, se zaměřením na kyčelní kloub.
7. Porovnejte délky vláken a ramena momentů sil s údaji v literatuře a kriticky zhodnoťte dosažené výsledky.

Rozsah diplomové práce:	doporuč. 50 s. původního textu
Rozsah grafických prací:	dle potřeby
Forma zpracování diplomové práce:	tištěná/elektronická
Jazyk zpracování:	Angličtina

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce:	doc. Ing. Josef Kohout, Ph.D. Nové technologie pro informační společnost
--------------------------	--

Datum zadání diplomové práce:	9. září 2024
Termín odevzdání diplomové práce:	14. května 2026

L.S.

doc. Ing. Miloš Železný, Ph.D.
děkan

doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 22. května 2025

Declaration

I hereby declare that this Master's Thesis is completely my own work and that I used only the cited sources, literature, and other resources. This thesis has not been used to obtain another or the same academic degree.

I acknowledge that my thesis is subject to the rights and obligations arising from Act No. 121/2000 Coll., the Copyright Act as amended, in particular the fact that the University of West Bohemia has the right to conclude a licence agreement for the use of this thesis as a school work pursuant to Section 60(1) of the Copyright Act.

During the preparation of this thesis, I have used Claude (Anthropic) for brainstorming ideas, improving the structure of the thesis, proofreading, code review, generation of Python scripts for data analysis and plotting, and assistance in locating relevant literature. After using this tool, I have carefully reviewed and edited all content as needed and take full responsibility for the resulting content of this work.

In Pilsen, on 13 May 2026

.....
Duc Long Hoang

The names of products, technologies, services, applications, companies, etc. used in the text may be trademarks or registered trademarks of their respective owners.

Abstract

This thesis extends the XPBD-based muscle deformation method within the Muscle Wrapping 2.0 framework by representing muscles as bundles of volume-invariant Cosserat rods. Each rod models a muscle-tendon unit and carries an additional degree of freedom that enforces local volume preservation during contraction. A novel slice distance constraint using nearest and furthest neighbor connections maintains inter-rod spacing without explicit collision detection. The method was validated on a synthetic arc experiment, and on a hip flexion simulation of four muscles, where the resulting moment arms align with published biomechanical data. The rod-based approach is approximately 24 times faster than the current surface-based method due to the embarrassingly parallel nature of per-rod constraint projection.

Abstrakt

Tato diplomová práce rozšiřuje metodu deformace svalů založenou na XPBD v rámci projektu Muscle Wrapping 2.0 o reprezentaci svalů jako svazky objemově nezávislých Cosseratových tyčí. Každá tyč modeluje svalovo-šlachovou jednotku a má o jeden stupeň volnosti navíc, který zajišťuje lokální zachování objemu během kontrakce. Nové omezení vzdálenosti v rámci řezu, které využívá nejbližší a nejvzdálenější sousední spojení, udržuje rozestupy mezi tyčemi bez nutnosti explicitní detekce kolizí. Metoda byla ověřena na syntetickém experimentu s obloukem a na simulaci čtyř svalů při flexi kyčle, kde výsledná ramena momentů jsou v souladu s publikovanými biomechanickými daty. Tento přístup založený na tyčích je přibližně 24krát rychlejší než metoda pracující s povrchem díky triviálně paralelní povaze projekce tyčových omezení.

Keywords

muscle deformation • Cosserat rod • volume preservation • XPBD • musculoskeletal simulation • parallelization

Acknowledgement

I would like to thank my thesis advisor, doc. Ing. Josef Kohout, Ph.D., for his guidance, thoughtful feedback, suggestions, and for the many consultations during which we even forgot to eat. I am grateful to Mgr. Eliška Haladová for her patience and support, especially during the final stages of the thesis. My thanks go to Mgr. Ondřej Komora and Ing. Martin Kukrál for their valuable help with the statistical theory, and to Bc. Jakub Šmrha for keeping me on track when I wanted to put this off. I also thank Ing. Kamil Ekštein, Ph.D., for the excellent $\LaTeX$ template, which was a blessing to work with. And finally, I would like to thank myself for not giving up and for the perseverance to finish this thesis, even when it was really hard.

Bc. Duc Long Hoang,
author of this Master's Thesis
(May 2026)

Contents

1	Introduction	5
2	Biological and anatomical background	7
2.1	Skeletal muscle architecture	7
2.1.1	Hierarchical organization	8
2.1.2	Fascicle arrangements	9
2.1.3	Physiological cross-sectional area	10
2.1.4	Attachment structures and muscle-tendon unit	11
2.2	Muscle contraction physiology	13
2.2.1	The sliding filament mechanism	13
2.2.2	Types of muscle contraction	14
2.2.3	Muscle roles during movement	15
2.3	Volume incompressibility	16
2.3.1	Physical basis of incompressibility	16
2.3.2	Implications for computational models	17
2.4	Biomechanical properties of interest	18
2.4.1	Moment arms	18
2.4.2	Force-length relationship	19
2.4.3	Fiber length changes during movement	19
3	Theoretical foundations	21
3.1	Cosserat rod theory	21
3.1.1	Geometric representation	21
3.1.2	Physical interpretation	22
3.1.3	Passive deformation and muscle activation	23
3.2	Volume invariant extension (VIPER)	23
3.2.1	Additional degree of freedom: radius	23
3.2.2	Volume preservation constraint	24
3.2.3	Physical interpretation	25
3.3	Discretization strategies	25
3.3.1	Centerline discretization	25

3.3.2	Local frame discretization	26
3.3.3	Radius discretization	28
4	Position-based dynamics framework	31
4.1	PBD fundamentals	31
4.1.1	Simulation overview	31
4.1.2	Constraint formulation	33
4.1.3	Constraint projection	33
4.1.4	Iterative solving	33
4.1.5	Stiffness control and its limitations	33
4.2	XPBD extension	34
4.2.1	Compliance-based formulation	34
4.2.2	Lagrange multiplier formulation	34
4.2.3	XPBD simulation loop	35
4.3	Comparison with standard PBD	36
5	Related work	39
5.1	Muscle Wrapping 2.0	39
5.1.1	Project objectives and scope	39
5.1.2	System architecture	40
5.2	Surface-based muscle deformation	41
5.2.1	PBD-based deformation	41
5.2.2	XPBD-based deformation	43
5.2.3	Limitations of surface-based approaches	44
5.3	Internal fiber generation methods	45
5.3.1	Template-based decomposition	45
5.3.2	Viperization	46
5.3.3	Point sampling based decomposition	47
5.4	Rod-based deformation in computer graphics	48
5.4.1	Cosserat rods in PBD	48
5.4.2	Volume Invariant Position-based Elastic Rods	49
5.4.3	Recent advances	50
5.5	Gap analysis and thesis contribution	50
5.5.1	Summary of existing approaches	50
5.5.2	Identified gap	51
5.5.3	Thesis contribution	51
6	Proposed method	53
6.1	Rod-based muscle representation	53
6.1.1	From fibers to rods	53

6.1.2	Discrete rod model	55
6.1.3	Rod bundle configuration	56
6.2	XPBD constraint formulation	56
6.2.1	Stretch-shear constraints	57
6.2.2	Bend-twist constraints	58
6.2.3	Volume preservation constraints	59
6.2.4	Attachment constraints	59
6.3	Collision handling	61
6.3.1	Rod-bone collision	61
6.3.2	Intra-bundle collision	62
6.4	Activation-driven contraction	63
6.4.1	Activation signal acquisition	63
6.4.2	Rest length modification	64
6.4.3	Contraction dynamics	65
6.5	System overview	66
6.5.1	Design goals and constraints	66
6.5.2	Integration with Muscle Wrapping 2.0	66
6.5.3	Simulation pipeline	67
7	Implementation details	69
7.1	Software architecture	69
7.1.1	Codebase organization	69
7.1.2	Source files	70
7.2	Integration with Muscle Wrapping 2.0	70
7.2.1	Choosing the integration approach	71
7.2.2	The VTK filter and internals pattern	71
7.2.3	Data layout in the solver	72
7.3	Rod data structures	72
7.3.1	From fibers to rods	72
7.3.2	The CosseratRod class	73
7.3.3	Initialization	73
7.4	Constraint implementation	74
7.4.1	Why standalone constraint classes	74
7.4.2	Stretch-shear constraint	75
7.4.3	Bend-twist constraint	75
7.4.4	Volume constraint	75
7.4.5	Slice distance constraint	76
7.4.6	Compliance time step convention	77
7.5	Collision handling	77
7.5.1	Intra-bundle collision	77

7.5.2	Rod-bone collision	77
7.5.3	Inter-muscular collision	79
7.6	Activation pipeline	79
7.6.1	Activation signal acquisition	79
7.6.2	Activation smoothing	79
7.6.3	Rest length modification	80
7.7	Solver loop	80
7.7.1	Extended solver loop	81
7.7.2	Constraint projection order	81
7.7.3	Compile-time feature flags	82
7.8	Parallelization	82
7.9	Validation infrastructure	84
7.9.1	Quarter-circle arc experiment	84
7.9.2	Statistical analysis pipeline	85
8	Results and evaluation	87
8.1	Experimental setup	87
8.2	Quarter-circle arc experiment	88
8.2.1	RQ1: Vertex distribution uniformity	88
8.2.2	RQ2: Shape preservation under activation	90
8.2.3	RQ3: Volume preservation	93
8.2.4	Summary of arc experiment results	95
8.3	Hip flexion experiment	96
8.3.1	Setup	96
8.3.2	Visual comparison: constraint contributions	96
8.3.3	Visual comparison: rod-based vs. surface-based	98
8.3.4	Fiber length comparison	99
8.3.5	Moment arm comparison	101
8.4	Performance analysis	104
9	Limitations and future work	107
10	Conclusion	109
	Bibliography	111
	List of Figures	115
	List of Tables	119
	List of Listings	121

Introduction

1

Modern medicine relies on computational models of musculoskeletal systems for a wide range of applications, including estimating contact forces on lower limb joints, simulating musculotendon mechanisms in both healthy and pathological individuals, and assisting medical professionals in surgical decision-making. The accuracy of these models is critical, as inaccurate representations can lead to incorrect decisions with potentially serious consequences for patients.

Current state-of-the-art muscle deformation models employ the Position Based Dynamics (PBD) method [Čer19] and its extended variant (XPBD) [Hav24] to simulate muscle behavior during skeletal motion. While these approaches have shown promise, they operate exclusively on the muscle surface geometry, omitting the internal structure of the muscle. Real skeletal muscles are composed of numerous muscle fibers arranged in specific architectures (e.g., parallel, pennate) whose collective contraction determines the overall muscle deformation. Furthermore, muscle tissue is nearly incompressible – when fibers shorten during contraction, the muscle must bulge to preserve volume. Surface-based models cannot inherently capture either the internal fiber behavior or this volume preservation, leading to unrealistic deformations. Havlíček [Hav24] observed that even with XPBD, improved collision handling, and physiologically-driven muscle activations, certain muscles (particularly the iliopsoas during extreme hip flexion) exhibit such artifacts.

Cosserat rod theory provides a mathematical framework for modeling one-dimensional elastic structures capable of bending, twisting, stretching, and shearing. Recent advancements, particularly the Volume Invariant Position-based Elastic Rods (VIPER) method [Ang+19], have extended classical Cosserat rods to maintain volume during deformation, which is crucial for realistic muscle simulation. By representing muscles as bundles of volume invariant Cosserat rods aligned with the physiological fiber directions, it may be possible to achieve more accurate and physiologically plausible muscle deformations.

This thesis investigates the integration of volume invariant Cosserat rods into an XPBD-based muscle deformation framework. By representing muscles as a collection of volume invariant elastic rods, the aim is to achieve more realistic muscle

deformations while maintaining the computational efficiency of PBD methods.

Biological and anatomical background

2

Realistic simulation of muscle deformation requires not only correct mathematical models but also a thorough understanding of the biological structures that are being modeled. This chapter provides the anatomical and physiological foundation necessary to understand why volume-preserving rod-based representations are suited for muscle simulation.

In biomechanical modeling, two distinct problems are commonly addressed:

1. **Forward dynamics** – muscle activations are given and the resulting forces and movement are computed
2. **Inverse dynamics** – observed bone motions are used to estimate the muscle states that produced them

This thesis focuses on the latter, simulating muscle deformation given kinematic input from tools such as OpenSim. In consequence, the biological detail presented here outlines structural and mechanical properties relevant to modeling, rather than delving into the neural pathways governing activation.

In the following sections, we examine the hierarchical organization of skeletal muscle tissue, the mechanisms of muscle contraction, muscle tissue incompressibility, and biomechanical properties relevant to computational modeling. This biological context establishes the requirements for the mathematical framework presented in Chapter 3.

2.1 Skeletal muscle architecture

Skeletal muscles are not homogenous masses of tissue but highly organized structures with a hierarchical arrangement. Understanding this architecture is crucial for

developing appropriate computational representations, as the internal organization of the tissue influences how forces are generated and propagated during movement.

2.1.1 Hierarchical organization

The structure of a skeletal muscle follows a nested hierarchy spanning several orders of magnitude, from the whole muscle to the molecular level responsible for contraction [Hav24; Gas+23]. As seen in Figure 2.1, a whole muscle is enclosed within a protective connective tissue called *epimysium*. Beneath this outer layer, the muscle is subdivided into bundles of muscle fibers called *fascicles*, each surrounded by another layer of connective tissue called *perimysium*.

Each fascicle comprises several tens of muscle fibers running in parallel [Hav24]. A muscle fiber is a single multinucleated cell (a *syncytium*), bounded by the *sarcolemma* membrane and contains thousands of rod-like structures called *myofibrils*. These myofibrils are also composed of smaller functional units called *sarcomeres*, which contain the *actin* (thin) and *myosin* (thick) protein filaments responsible for generating contractile forces.

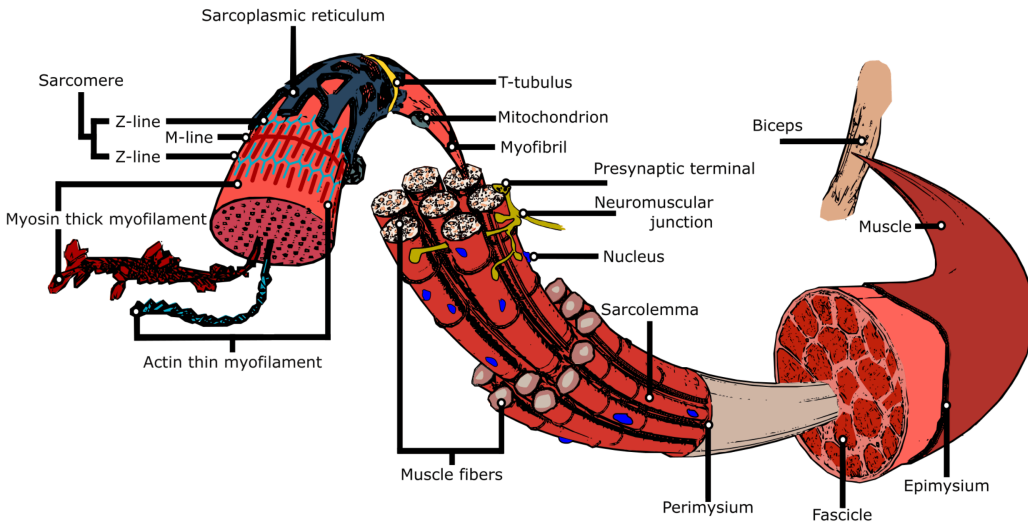


Figure 2.1: The hierarchical structure of skeletal muscle, from the whole muscle down to the sarcomere. Image from [Hav24]

This hierarchical organization has important implications for computational modeling. While simulating individual sarcomeres or even myofibrils would be computationally excessive and disproportionate, the fascicle level becomes an acceptable compromise that provides a natural abstraction for rod-based representations. As noted by Angles et al. [Ang+19], fascicles can be abstracted into a collection of elastic rods, with each rod representing the mechanical behavior of a bundle of muscle fibers.

2.1.2 Fascicle arrangements

While fibers within a fascicle always run parallel to each other, the fascicles themselves can be organized in various arrangements that significantly affect the muscle's mechanical properties [Bet+13]. Various fascicle arrangements are visualized in Figure 2.2.

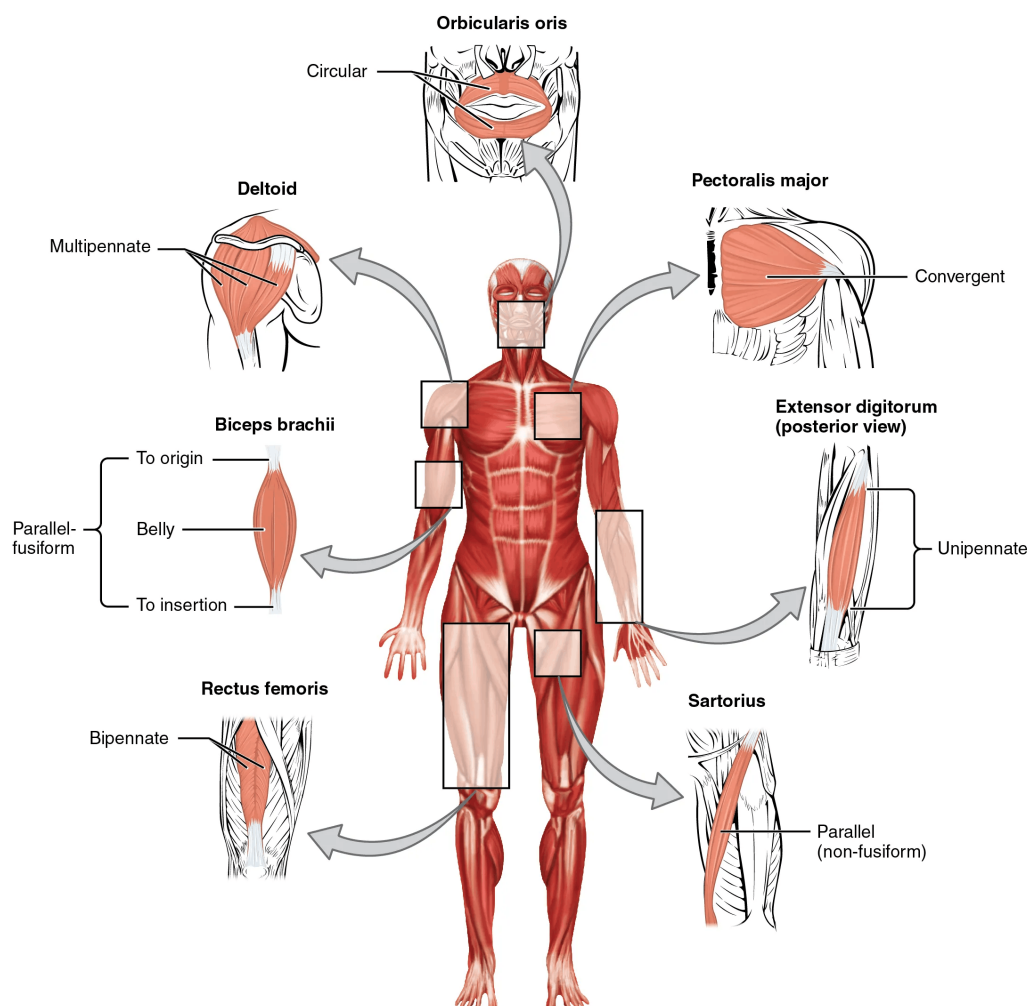


Figure 2.2: Common fascicle arrangements in skeletal muscle. Image from [Bet+13]

The most notable fascicle arrangements include:

- **Parallel (fusiform)** – fascicles in the same direction as the long axis of the muscle. See *sartorius* in Figure 2.2
- **Circular** – fascicles arranged in concentric rings, e.g., *orbicularis oris*
- **Convergent** – fascicles expand over a wide area and converge to a single, common attachment area, e.g., *pectoralis major*

- **Pennate** – fascicles attaching to a tendon at an angle (*pennation angle*) allow for it to contain more muscle fibers. This enables greater force production, though at a reduced range of motion. Sub-categories include *unipennate* (e.g., *extensor digitorum*), *bipennate* (e.g., *rectus femoris*) and *multipennate* (e.g., *deltoid*).

These fascicle arrangements can be represented computationally using predefined fiber templates, as described by Blemker and Delp [BD05] and implemented by Kohout and Kukačka [KK14]. Figure 2.3 illustrates the parallel and pennate templates used in the Muscle Wrapping 2.0 framework, where fibers are expressed analytically via rational Bézier spline curves.

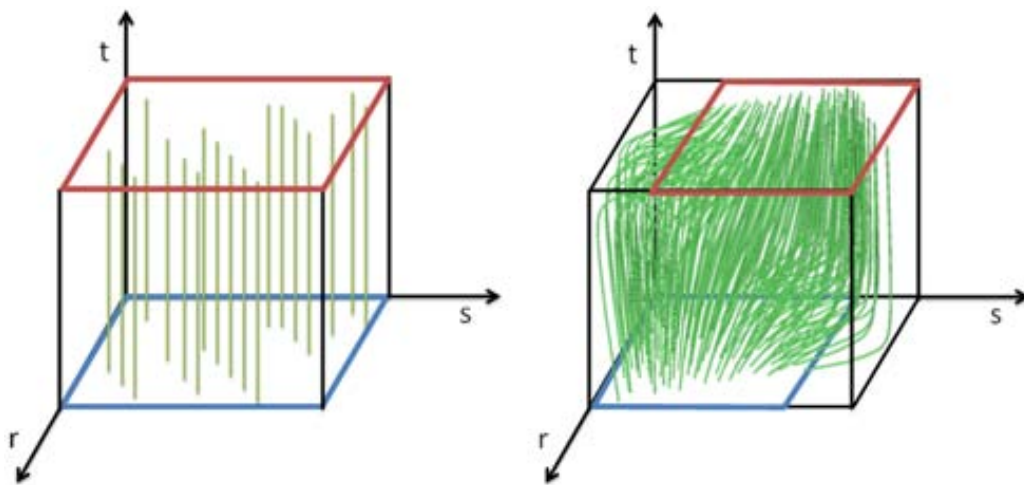


Figure 2.3: Predefined fiber templates: parallel (left) and pennate (right). Attachment areas are highlighted in red and blue. Image from [KK14]

2.1.3 Physiological cross-sectional area

The maximum force a muscle can generate is not simply proportional to its *anatomical cross-sectional area* (ACSA) – the area perpendicular to the muscle’s long axis, but rather to its *physiological cross-sectional area* (PCSA). The PCSA takes into account the pennation angle and fiber length, explaining the greater force generation when compared to the parallel fascicle arrangement despite the shorter fibers, as stated in Subsection 2.1.2. The difference between ACSA and PCSA of a pennate arrangement is illustrated in Figure 2.4.

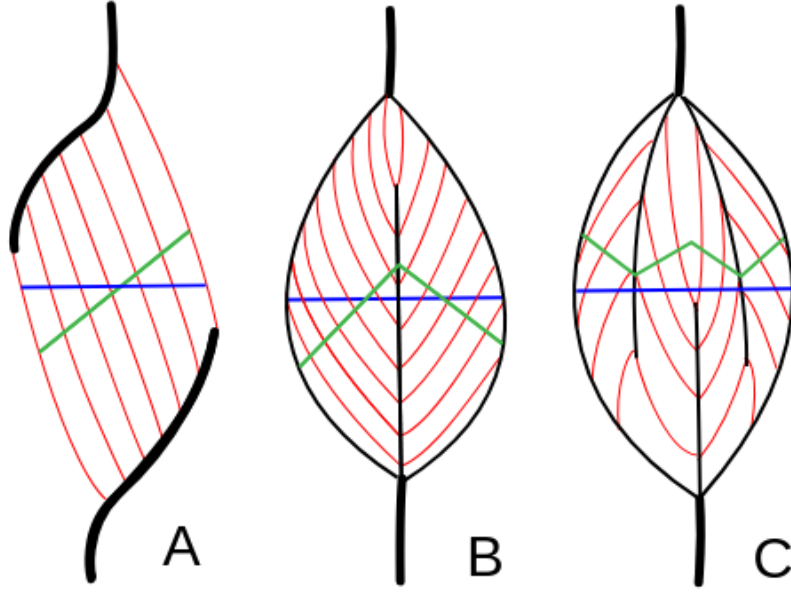


Figure 2.4: Pennate muscle fiber arrangements. PCSA in green, ACSA in blue. Image from [Wik07]

The PCSA is calculated as follows [Mar+20]:

$$\text{PCSA (cm}^2\text{)} = \frac{\text{muscle mass (g)} \times \text{pennation angle } (\cos \alpha)}{\text{muscle density (g cm}^{-3}\text{)} \times \text{fascicle length (cm)}}, \quad (2.1)$$

where α is the pennation angle. When computing forces transferred to bones, biomechanical models must account for both the pennation angle and PCSA, even when the muscle itself is represented by simplified geometric primitives.

2.1.4 Attachment structures and muscle-tendon unit

Muscles attach to bones through various connective tissues including tendons, fascia, and aponeuroses at two primary locations: the *origin* (typically the more fixed attachment) and the *insertion* (usually the more mobile attachment) [Bet+13]. Muscles with more complex architectures, like *gluteus maximus*, feature multiple origin and/or insertion areas.

These attachment areas serve as boundary constraints in fiber-based deformation models. In the Muscle Wrapping 2.0 framework, attachment areas are defined by expert-specified landmarks (points of interest) and remain fixed to the underlying bone during simulations [Hoa23; KK14]. The fiber endpoints remain anchored to these attachment areas while the rest of the fiber structure deforms in accordance with the applied forces and constraints. Figure 2.5 depicts the *gluteus medius* and *iliacus* muscles with their attachment areas.

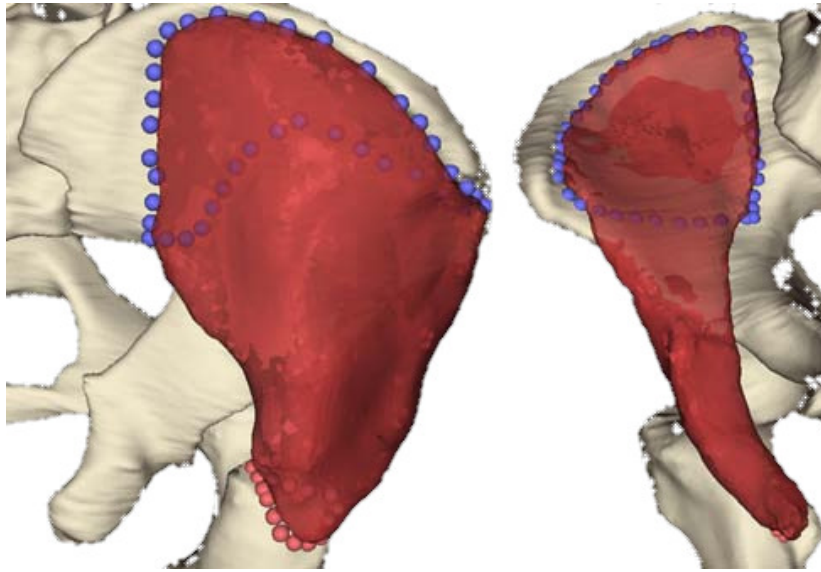


Figure 2.5: *Gluteus medius* (left) and *iliacus* (right) with their attachment areas. Blue landmarks identify the origin area, while the red landmarks correspond to the insertion area. Image from [KK14]

For biomechanical modeling purposes, the relevant functional unit is not the muscle belly alone but the *muscle-tendon unit* (MTU) – the complete structure from one attachment to another, including the muscle belly, aponeuroses, and tendons. What is referred to as ‘muscle’ in the context of computational models typically encompasses the entire MTU. In the musculoskeletal models considered in this thesis, muscles are always connected to bones at both ends.

This distinction is important as it justifies the rod-based representation used in this thesis. Modeling a muscle as one or more rods connecting attachment points on two bones is not a novel concept but rather a valid approach for the vast majority of skeletal muscles [KK14; Arn+09; BD05]. For muscles with wide attachment areas, multiple rods are used to capture the force distribution. Kohout and Kukačka [KK14] address this by generating parallel fibers throughout the entire muscle-tendon unit volume, respecting the specified attachment areas. For muscles where pennation angle $\alpha \approx 0$, this provides a good modeling approximation.

It should be noted that muscles with particularly complex architectures, such as *rectus femoris* and *soleus*, may require more sophisticated representations. Nevertheless, even simplified rod-based models provide improvements over traditional line-segment approaches (e.g., Delp [BD05] and Arnold [Arn+09] models) that represent such muscles with a single line of action.

Having established the structural organization of skeletal muscles, the next focus should be the physiological mechanisms by which these structures generate force and initiate movement.

2.2 Muscle contraction physiology

This section examines the physiological mechanisms by which skeletal muscles create force and produce movement. Since this thesis addresses the inverse dynamics problem, simulating muscle deformation given bone kinematics, the focus is on the mechanical aspects of contraction rather than neural signaling pathways.

2.2.1 The sliding filament mechanism

Muscle contraction occurs at the sarcomere level where interlaced thick (myosin) and thin (actin) filaments start sliding against each other without changing their individual lengths. This is known as the *sliding filament theory*, first described independently by the two pairs Hugh Huxley & Jean Hanson and Andrew Huxley & Rolf Niedergerke in 1954.

When a muscle is activated, the *myosin* heads form cross-bridges with *actin* filaments and perform repeated power strokes, pulling the thin filaments toward the center of the sarcomere. This shortens the sarcomere length, leading to overall muscle contraction. A sarcomere can typically contract to 50–70 % of its resting length [Hav24]. The contraction cycle is visualized in Figure 2.6.

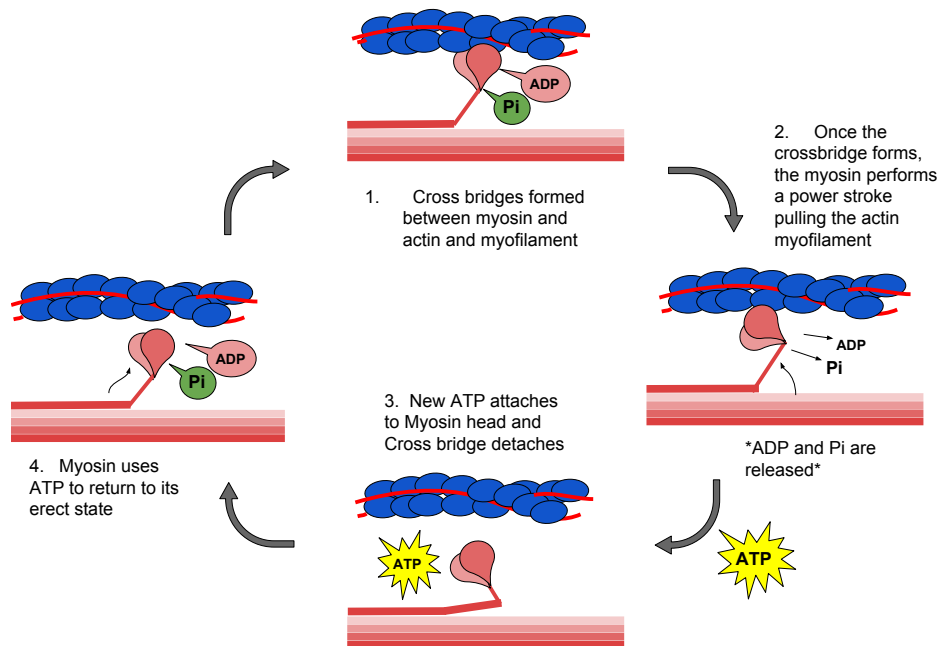


Figure 2.6: The sliding filament mechanism showing cross-bridge formation and power stroke. Image from [Wik23]

This shortening at the sarcomere level propagates through the hierarchical structure, sarcomeres shorten myofibrils, myofibrils shorten muscle fibers, and muscle

fibers shorten fascicles, ultimately leading to the contraction of the entire muscle. Throughout this process, the muscle tissue maintains approximately constant volume due to its near-incompressibility (see Section 2.3).

2.2.2 Types of muscle contraction

Muscle contractions can be divided based on the relationship between muscle tension and changes in muscle length [Gas+23] – see Figure 2.7.

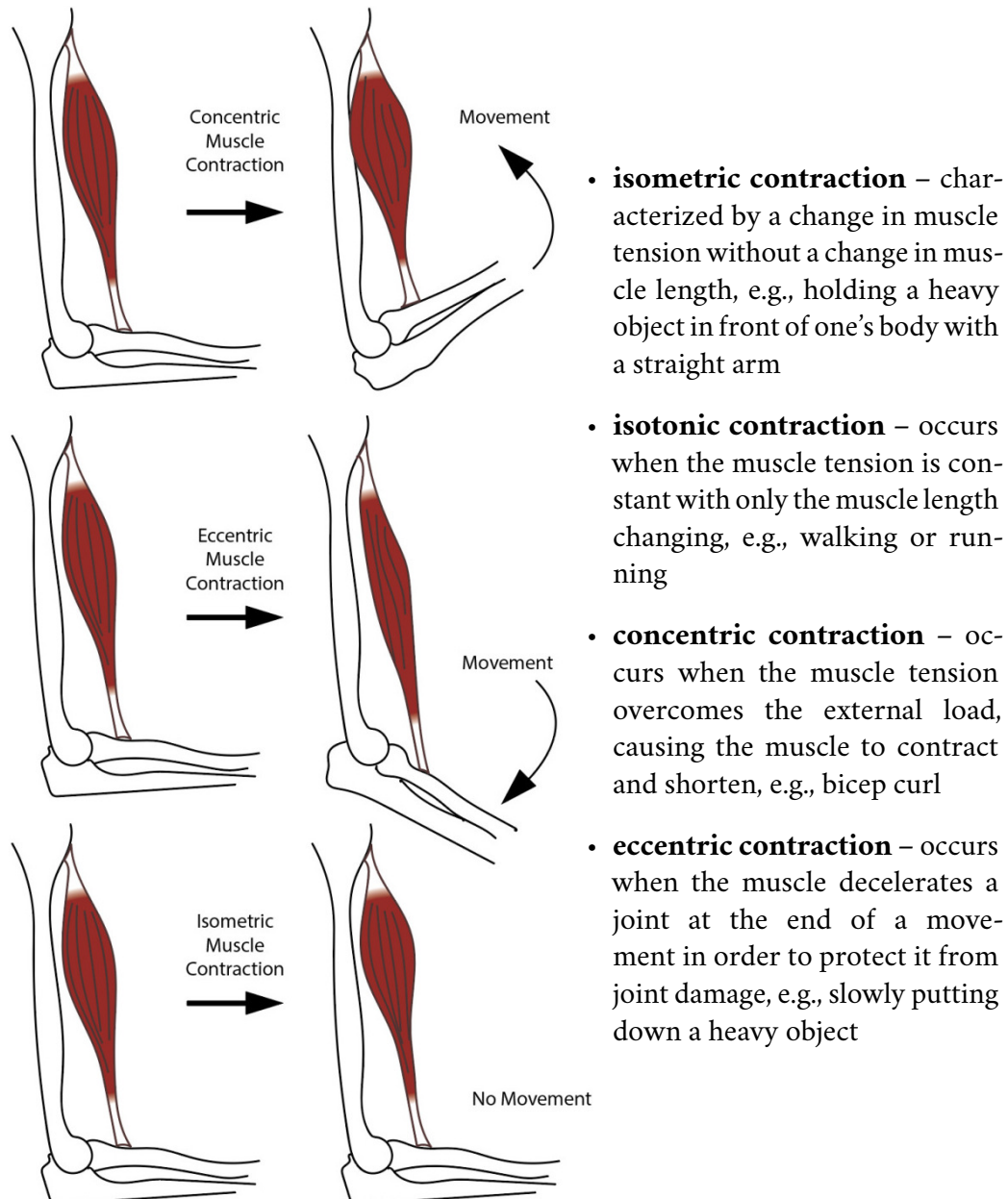


Figure 2.7: Types of muscle contraction. Concentric (top), eccentric (middle) and isometric (bottom). Image from [Gas+23]

In computational models, the difference between the contraction types is important because the deformation of each results in a different pattern.

2.2.3 Muscle roles during movement

Complex movements require multiple muscles to act in coordination, each in a different role [Bet+13]:

- **agonist (prime mover)** – the principal muscle responsible for movement
- **antagonist** – the muscle with the opposite action of the prime mover, usually located on the opposite side of the bone
- **synergist** – a muscle that assists the prime mover during the motion
- **fixator** – a special type of a synergist muscle that stabilizes the bone where the agonist has its origin area

For example, during elbow flexion (see Figure 2.8), the *biceps brachii* assumes the role of the agonist, while the *triceps brachii* acts as the antagonist. The *brachioradialis* and *brachialis* serve as the synergists, aiding with the movement. The *deltoid* acts as a fixator stabilizing the shoulder.

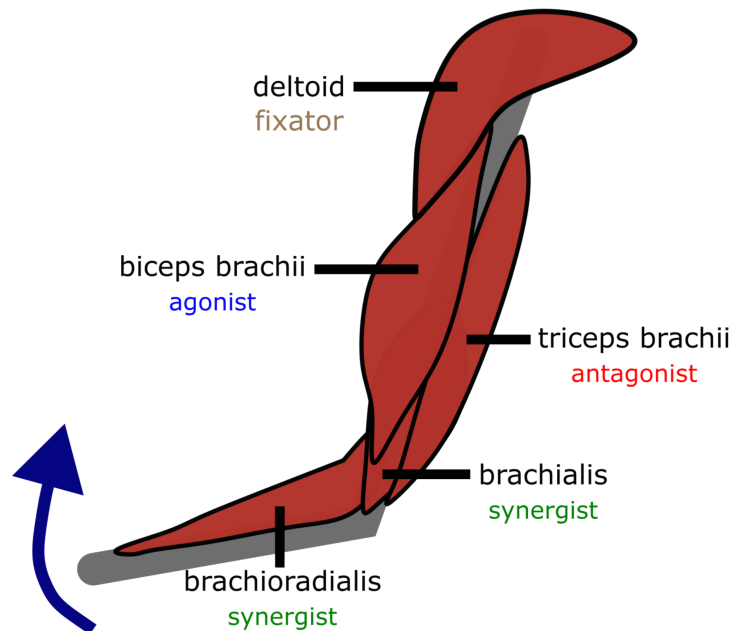


Figure 2.8: Muscle roles during elbow flexion. Image from [Hav24]

This coordinated action is important for muscle simulation. When modeling a movement sequence, multiple muscle activations need to be taken into account

rather than focusing on each muscle individually. Simulation tools like OpenSim provide the necessary framework to configure the activation levels of each muscle during movement.

2.3 Volume incompressibility

One of the most important mechanical properties of muscle tissue for computational modeling (and more so for this thesis) is its near-incompressibility. This section will explore the physical basis for said property and the implications for muscle simulation.

2.3.1 Physical basis of incompressibility

Total body water comprises $\approx 45\text{--}75\%$ of an individual's body weight, and skeletal muscle tissue has a water content typically around $70\text{--}75\%$ by mass [RD13]. The high water content in conjunction with the many layers of connective tissue (*epimysium*, *perimysium*, etc.) allows the muscle tissue to behave as a nearly incompressible material.

The incompressibility of a material is given by its Poisson's ratio ν , a measure of the Poisson effect. This effect describes the deformation of a material in directions perpendicular to the direction of the load – see Figure 2.9.

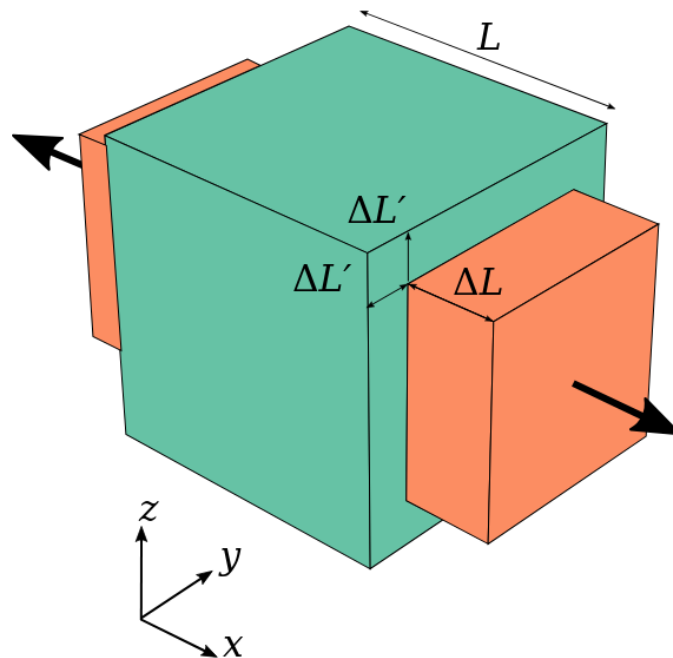


Figure 2.9: Illustration of the Poisson effect before (green) and after (red) an object is stretched. The load in the longitudinal axis x elicits a response in both the lateral axes, y and z . Image from [Wik24]

The Poisson's ratio is calculated as the negative ratio of the transverse (lateral) strain to the axial (longitudinal) strain:

$$\nu = -\frac{d\epsilon_{transverse}}{d\epsilon_{axial}} \quad (2.2)$$

A typical value of ν for most solids is between 0 and 0.5, with a perfectly incompressible material having $\nu = 0.5$, meaning that compression in one direction is equally compensated by expansion in perpendicular directions. This phenomenon leads to the preservation of volume. Past experimental measurements concluded that skeletal muscle has a Poisson's ratio in the range of 0.47–0.49 [Tak+13; Ang+19], nearing the incompressibility limit.

This volume preservation behavior is a fundamental constraint that must be adhered to in realistic muscle simulations.

2.3.2 Implications for computational models

Traditional surface-based deformation models (i.e., based on mesh skinning or basic Position-Based Dynamics) do not innately enforce volume preservation. This can lead to unrealistic deformations where muscles shrink or expand without any physical cause. Well-known problems include the *collapsing joint* and the *candy wrapper* effect [FTS08] – see Figure 2.10.

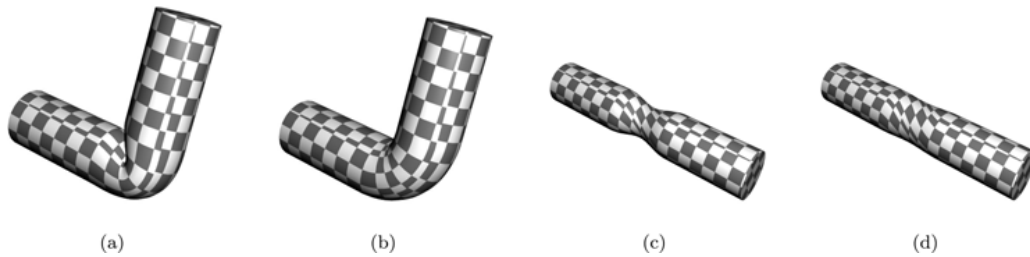


Figure 2.10: Subfigures (a) and (c) show the collapsing joint and candy wrapper effects when an object bends and twists respectively. Subfigures (b) and (d) are free of these phenomena. Image from [MMG06]

Several approaches have been developed to address this limitation:

- **Volume constraints in PBD** – achievable via explicit constraints that preserve the volume of tetrahedral elements or the global mesh volume. However, this can be computationally expensive and may not capture the local volume preservation behavior of the fibrous structures correctly.
- **Finite Elements Method with incompressible materials** – FEM can model muscle tissue with constitutive models with high bulk modulus or explicit incompressibility constraints. Angles et al. [Ang+19] compared their rod-based

model to a tetrahedral FEM discretization with a hyperelastic Neo-Hookean model using a Poisson's ratio $\nu = 0.49$, which demonstrated similar volume-preserving behavior.

- **Volume-invariant rod models** – the VIPER approach [Ang+19] directly integrates volume preservation into the rod formulation by utilizing the cross-sectional radius as another degree of freedom. This additional degree of freedom is designated to maintain constant volume of the rod as it stretches and compresses.

The volume-invariant rod approach (VIPER) is especially suited for muscle modeling as it blends computational efficiency of rod representations with explicit volume preservation. However, it is important to note that VIPER decomposes muscles into fibers (rods) and therefore

$$\sum_{i=1}^n V_{rod_i} \neq V_{muscle}.$$

As a consequence, the reconstruction of the entire muscle from the rods may not return the same volume as the initial input.

2.4 Biomechanical properties of interest

Alongside the structural and physiological aspects already described, there are other biomechanical properties of importance for validating computational muscle models. This section briefly discusses the key metrics used to evaluate the realism of muscle simulations.

2.4.1 Moment arms

The moment arm of a muscle is the perpendicular distance from the joint's center of rotation to the line of action of the muscle's force [SSD13]. It determines how effectively a muscle produces rotation at the joint (torque), with a longer moment arm resulting in greater mechanical advantage, allowing it to create more torque with less force.

In computational models, moment arms are crucial for the following reasons:

- They vary significantly across fibers within the same muscle, this is true especially for muscles with complex geometries and architectures [BD05].
- They change as the joint moves through its range of motion.
- They serve as an objective validation metric when comparing computed moment arms with experimental measurements.

2.4.2 Force-length relationship

The force a muscle can generate is directly tied to its current length relative to its optimal length. The force-length relationship describes this dependency, showing that the maximum force is produced when the muscle is at its optimal length, with reduced force generation at shorter and longer lengths [BD05; MLH20].

This relationship arises from the sarcomere mechanics (described in Subsection 2.2.1): at the optimal length, the overlap between *actin* and *myosin* filaments allows for maximum cross-bridge formation. At shorter lengths, the filaments interfere with each other, while at longer lengths fewer cross-bridges are formed due to reduced overlap.

For rod-based simulations, it is important that the simulated contractions show close correspondence with physiological expectations.

2.4.3 Fiber length changes during movement

During joint movement, muscle fibers undergo a change in their length – this is also known as ***fiber excursions***. These fiber excursions can be predicted with the help of computational models and validated against experimentally measured data from static MRI [BD05; MK20].

Validating fiber excursions provides a more relevant assessment of a model's accuracy than moment arms alone and a model that correctly predicts both has successfully captured the underlying muscle geometry and mechanics.

Summary

This chapter has provided the biological and anatomical foundation necessary to understand the requirements for realistic muscle simulation. The most important factors can be summarized as follows:

- Skeletal muscles demonstrate a hierarchical structure from the whole muscle down to the sarcomeres, with the fascicle level providing an appropriate abstraction for rod-based computational models.
- Fascicle arrangements (parallel, pennate, etc.) affect the mechanical properties of a muscle and can be represented via predefined fiber templates.
- Muscle contraction occurs through the sliding filament mechanism where actin and myosin filaments interact to shorten sarcomeres to 50–70 % of their resting length.

- Muscle tissue is nearly incompressible due to its high water content (Poisson's ratio ≈ 0.5).
- Biomechanical properties (e.g., moment arms) provide a means of validation and scoring for computational models.

These biological facts motivate the use of Cosserat rods for muscle simulation and in particular the VIPER formulation for volume preservation. The following Chapter 3 describes the mathematical framework in more detail.

Theoretical foundations

3

Realistic simulation of muscle deformation requires a mathematical model capable of capturing the complex mechanical behavior of fibers during bending, twisting, stretching, and shearing. This chapter provides the theoretical foundation for the proposed approach by presenting the *Cosserat rod theory*, a well-established model for simulating slender elastic structures. In the following sections, the volume-invariant extension (VIPER) of Cosserat rods is introduced, which adds an additional constraint to enforce volume preservation. Lastly, discretization strategies are discussed for computational implementation within a position-based dynamics framework.

3.1 Cosserat rod theory

The Cosserat rod theory, developed by the Cosserat brothers in 1909 [CC09] and first implemented in computer graphics by Pai in 2002 [Pai02], is a generalization of the Kirchhoff rod theory which models 1-D slender rods capable of only bending and twisting [Zha+19; Gaz+18]. Cosserat rods add the ability to stretch and shear, allowing for all possible forms of deformation thanks to the representation of both the centerline geometry and the orientation of the rod's cross-sections along its length. One key assumption is that the rod's length is much larger than its radius ($L \gg r$), allowing the dynamical behavior to be approximated by averaging balance laws over the cross-sections.

3.1.1 Geometric representation

A Cosserat rod (see Figure 3.1) is represented by:

- **Centerline:** $\mathbf{c}(s, t)$, where $s \in [0, L]$ is the arc-length of the rod and t is time
- **Local reference frame:** $\mathbf{R}(s, t) = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3]$, a rotation matrix whose columns are the orthonormal basis vectors (*directors*) attached to each cross-section of the rod at point s .

A convention is to define a laboratory (Eulerian) frame with orthonormal basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and a local (Lagrangian) frame spanned by the directors $\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3\}$ at each cross-section. Any direction vector $\mathbf{v}$ can be expressed in either basis:

$$\mathbf{v} = \bar{v}_1 \mathbf{e}_1 + \bar{v}_2 \mathbf{e}_2 + \bar{v}_3 \mathbf{e}_3, \quad (3.1)$$

$$\mathbf{v} = v_1 \mathbf{d}_1 + v_2 \mathbf{d}_2 + v_3 \mathbf{d}_3. \quad (3.2)$$

The components in the two frames are related by the rotation matrix $\mathbf{R} = [\mathbf{d}_1 \ \mathbf{d}_2 \ \mathbf{d}_3]$:

$$\bar{\mathbf{v}} = \mathbf{R} \mathbf{v}_{\text{local}} \quad \text{and} \quad \mathbf{v}_{\text{local}} = \mathbf{R}^T \bar{\mathbf{v}} \quad (3.3)$$

For transforming point coordinates between the world frame and the local frame anchored at the centerline point $\mathbf{c}(s)$, the full transformation includes both rotation and translation:

$$\bar{\mathbf{p}} = \mathbf{c}(s) + \mathbf{R}(s) \mathbf{p}_{\text{local}} \quad (3.4)$$

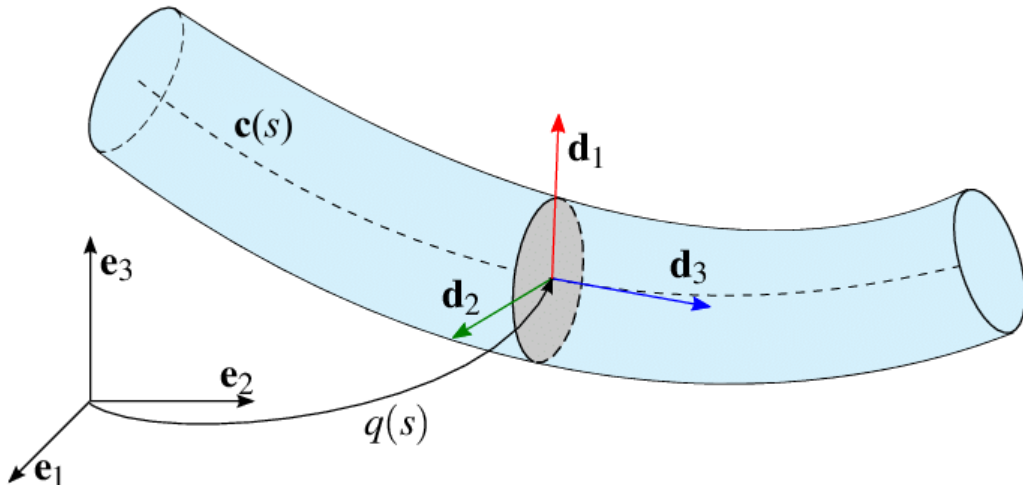


Figure 3.1: Cosserat rod with the triad $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ representing the Eulerian frame while the vectors $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$ describe the local reference frame. Adapted from [KS16]

3.1.2 Physical interpretation

The Cosserat rod theory allows for a complete characterization of rod mechanics through six degrees of freedom at each point: 3 translational – to describe the location of a point, and 3 rotational – to capture the orientation of an associated cross-section.

For muscle fiber simulation, Cosserat rods provide the necessary expressiveness to capture the complex deformations that occur during movement.

However, classical Cosserat rods have a fundamental limitation for muscle modeling, as they do not enforce volume conservation. As discussed in Section 2.3, real

muscle tissue is nearly incompressible, and therefore deformations must preserve volume. The next section presents a volume-invariant extension that addresses this limitation.

3.1.3 Passive deformation and muscle activation

It is important to note that the Cosserat rod theory describes passive elastic structures that deform in response to external forces but do not autonomously generate internal forces. This contrasts with biological muscles, which are active tissues capable of generating forces through contraction. The classical Cosserat rod theory completely neglects this significant aspect. However, it is still used in practice as demonstrated by [Ang+19].

To use Cosserat rods for muscle simulation, additional mechanisms must be introduced to represent physiological behavior e.g., external forces, collision handling, and constraints that mimic muscle contraction.

3.2 Volume invariant extension (VIPER)

Classical Cosserat rods allow arbitrary stretch and compression along their length, which can lead to changes in volume during deformation. This is physiologically inaccurate, as muscle tissue is approximately 75 % water with a Poisson's ratio of 0.5 [Tak+13], indicating near incompressibility. To address this shortcoming, Angles et al. [Ang+19] introduced the **Volume Invariant Position-based Elastic Rods** (VIPER) method, which adds an additional constraint to the Cosserat rod for enforcing volume preservation.

3.2.1 Additional degree of freedom: radius

The key idea of VIPER is the introduction of an additional degree of freedom at each cross-section of the rod. The classical Cosserat rod model is therefore generalized to include a variable radius $r(s)$ along the rod, previously being constant.

The VIPER rod at each point $s \in [0, L]$ is thus represented by:

- **Centerline:** $\mathbf{c}(s, t)$
- **Local reference frame:** $\mathbf{R}(s, t) = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3]$
- **Radius:** $r(s) \in \mathbb{R}^+$

This additional degree of freedom enables the rod to satisfy the incompressibility constraint: when the rod stretches, the radius decreases accordingly to maintain constant volume, and vice versa.

Figure 3.2 illustrates the difference between classical Cosserat rods and VIPER rods under stretching and compression. Cosserat rods maintain a constant radius, resulting in volume change, while VIPER rods adjust their radii to preserve volume. When stretched, the rod becomes thinner, and when compressed, it bulges.

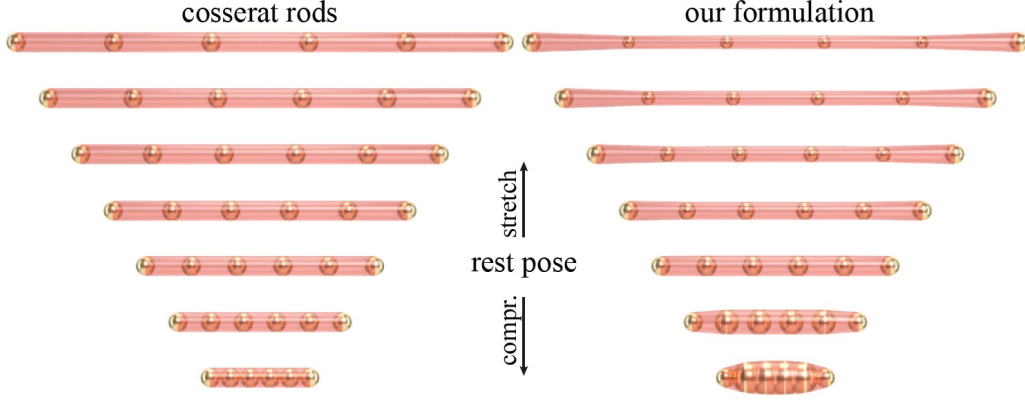


Figure 3.2: Comparison between classical Cosserat rods (left) and VIPER rods (right) under the effects of stretching and compression. Image from [Ang+19]

3.2.2 Volume preservation constraint

For a rod segment of infinitesimal length ds , the volume element dV is given by:

$$dV = \pi r(s)^2 ds \quad (3.5)$$

In the rest configuration with radius $r_0(s)$ and arc-length element ds_0 , the volume is:

$$dV_0 = \pi r_0(s)^2 ds_0 \quad (3.6)$$

The volume preservation constraint requires that these volumes remain equal during deformation:

$$\pi r(s)^2 ds = \pi r_0(s)^2 ds_0 \quad (3.7)$$

Rearranging this equation gives the relationship between the current and rest radius:

$$r(s) = r_0(s) \sqrt{\frac{ds_0}{ds}} \quad (3.8)$$

This equation captures the essential behavior: when a rod segment shortens ($ds < ds_0$), the term $\sqrt{\frac{ds_0}{ds}} > 1$, meaning the radius must increase ($r > r_0$), which produces the bulging effect. Conversely, when the segment stretches ($ds > ds_0$), the radius decreases ($r < r_0$), resulting in thinning.

3.2.3 Physical interpretation

For muscle fiber simulation, VIPER rods provide a more accurate representation of muscle mechanics than classical Cosserat rods. At each point along the rod there are now 7 degrees of freedom (3 translational, 3 rotational, and 1 radial), allowing the rod to capture complex deformations while preserving volume.

However, to implement the volume invariant Cosserat rod model within a computational framework, the continuous rod model must be discretized into a finite set of segments suited for numerical simulations. The next section discusses discretization strategies for these rods.

3.3 Discretization strategies

The continuous Cosserat rod model presented in the previous sections must be discretized for computational implementation. This section describes how the centerline, local reference frames, and radii can be approximated using a finite number of elements suitable for numerical simulations.

3.3.1 Centerline discretization

The continuous centerline $\mathbf{c}(s, t)$ is approximated as a polyline consisting of $N + 1$ vertices $\{p_0, p_1, \dots, p_N\}$ and N edges $\{1, \dots, N\}$, where each edge e_i connects vertices p_{i-1} and p_i [USS14; KS16] – see Figure 3.3.

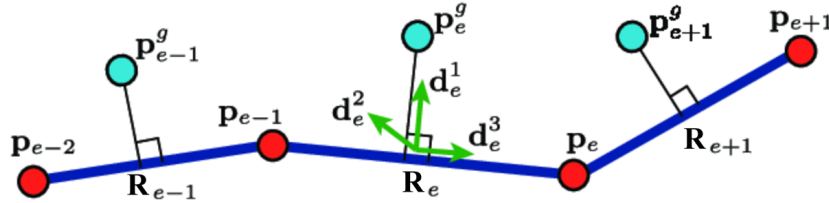


Figure 3.3: Discretization of the centerline into a series of connected line segments. Vertices in red, edges in dark blue. Adapted from [USS14]

For each edge e , we define:

- **Current length:** $l_e = |p_e - p_{e-1}|$
- **Reference length:** $\hat{l}_e = |p_e - p_{e-1}|_0$
- **Tangent direction:** $t_e = \frac{p_e - p_{e-1}}{l_e}$

This approximation allows for efficient computation while maintaining sufficient accuracy for simulating rod dynamics.

3.3.2 Local frame discretization

Local reference frames $\mathbf{R}(s, t)$ must be defined at discrete locations along the rod – see Figure 3.4.

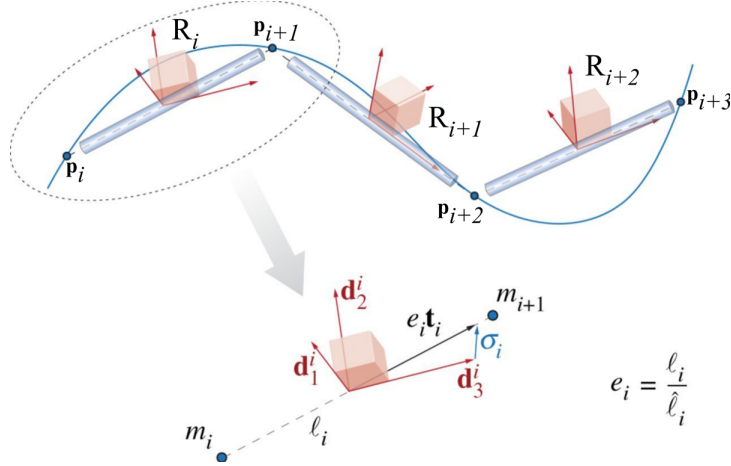


Figure 3.4: Discretization of a continuous Cosserat rod into segments with an associated local reference frame $\mathbf{R}_i$ for orientation and an edge defined by two vertices $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. Adapted from [Gaz+18]

There exist two common approaches for defining reference frames and their orientations:

1. **Ghost points** – due to the constraint of PBD working only with particles, Umetani et al. [USS14] defined auxiliary ghost points to represent the frames of each segment. The ghost particles are positioned perpendicular to each edge and constraints are used to force them to move along with the curve during the simulation – ghost points p^g are shown in cyan in Figure 3.3
2. **Quaternions** – Kugelstadt et al. [KS16] enhanced the PBD framework to support the definition and constraint solving between orthonormal frames directly. This was achieved by representing the frames as unit quaternions – see Figure 3.5

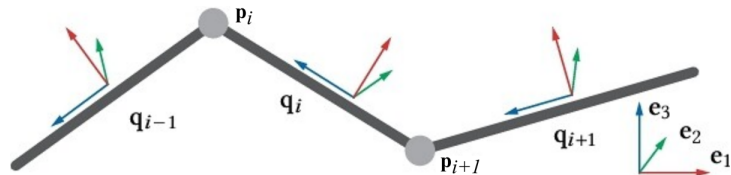


Figure 3.5: Reference frames represented as unit quaternions $\mathbf{q}_i$ at each segment of the discretized rod. Adapted from [Hsu+25]

3.3.2.1 Approaches to orientation representation

For this thesis, the orientation of frames will be represented using quaternions due to their computational efficiency and avoidance of gimbal lock. To understand this choice, quaternions and other common methods for representing orientation will be briefly discussed:

Euler angles. Euler angles describe the orientation of an object in three-dimensional space with the help of three angles – roll ϕ , pitch θ and yaw ψ by which the object is rotated around the x , y and z axes respectively. The rotations around each of these axes can be represented by the following matrices:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

However, in practice they are substituted by one rotation matrix $\mathbf{R}$ as follows:

$$\mathbf{R} = R_x(\phi)R_y(\theta)R_z(\psi)$$

The main advantage of Euler angles is the intuitive and simple nature of the representation, however they suffer from numerous drawbacks. One of them is the gimbal lock phenomenon, which occurs when two of the three rotational axes align, resulting in a loss of one degree of freedom. This can lead to unexpected behavior during simulations and adds an unnecessary overhead during implementation.

Another disadvantage is the computational complexity of converting between Euler angles and rotation matrices as well as costly matrix multiplications.

Axis angle (Rodrigues' rotation formula). This method represents rotation using a unit vector $\mathbf{u} = (u_x, u_y, u_z)$ – the axis of rotation, and an angle θ – the magnitude of rotation around axis $\mathbf{u}$. Rotating a vector $\mathbf{v}$ by angle θ around axis $\mathbf{u}$ is given by the Rodrigues' rotation formula:

$$\mathbf{v}_{rot} = \mathbf{v} \cos(\theta) + (\mathbf{u} \times \mathbf{v}) \sin(\theta) + \mathbf{u}(\mathbf{u} \cdot \mathbf{v})(1 - \cos(\theta)) \quad (3.9)$$

This representation is computationally efficient and avoids gimbal lock, however it is less intuitive than Euler angles and multiple rotations cannot be easily combined as with matrix multiplication.

Quaternions. Quaternions can be imagined as an extension of complex numbers into four dimensions. A quaternion $\mathbf{q}$ is represented as:

$$q = w + xi + yj + zk \quad w, x, y, z \in \mathbb{R}, \quad (3.10)$$

where i, j and k are imaginary units that satisfy the following multiplication rules:

$$i^2 = j^2 = k^2 = ijk = -1. \quad (3.11)$$

They are used in gyroscopes of modern smartphones and in 3D computer graphics due to their numerous advantages:

- **No gimbal lock** – quaternions cover the entire 3D rotation space without any singularities
- **Compact** – only 4 values are needed to represent a rotation, compared to 9 in a rotation matrix
- **Efficient composition** – combining rotations is done via quaternion multiplication, a much cheaper operation than matrix multiplication
- **Smooth interpolation (SLERP)** – quaternions allow for smooth spherical linear interpolation between orientations
- **PBD integration support** – Kugelstadt et al. [KS16] successfully demonstrated the usage of quaternions within a constraint-based simulation framework

Given these advantages, quaternions are the preferred choice for representing orientations of reference frames in this thesis.

3.3.3 Radius discretization

For the VIPER rod model, the continuously varying radius $r(s, t)$ introduced in Section 3.2.1 must also be discretized. Contrary to the centerline and orientations which have established discretization methods, the radius can be discretized in multiple ways depending on the desired accuracy and implementation complexity.

3.3.3.1 Vertex-based radius

The original VIPER paper [Ang+19] assigns a radius value $r_i \in \mathbb{R}^+$ to each vertex p_i as seen in Figure 3.6.



Figure 3.6: From left to right: Illustration of the pectoral, biceps and deltoid muscles and the representation of their radii at vertices. Image from [Ang+19]

For a rod of N edges and $N + 1$ vertices, this results in $N + 1$ radius values. Each edge e_i connected by vertices p_{i-1} and p_i then forms a **truncated cone** (frustum) with radii r_{i-1} and r_i at its ends – see Figure 3.7.

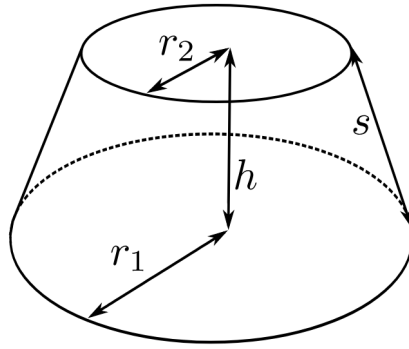


Figure 3.7: A truncated cone (frustum) with radii r_1 and r_2 at its ends and height h . Image from [Wik08]

The volume of a frustum is given by:

$$V = \frac{1}{3}\pi h(r_1^2 + r_1r_2 + r_2^2). \quad (3.12)$$

This approach provides smooth radius transitions along the rod and looks more visually appealing. However, it introduces additional complexity in enforcing the volume preservation as each rod segment now depends on two radius values.

3.3.3.2 Edge-based radius

A simpler alternative is to assign a single radius value $r_e \in \mathbb{R}^+$ to each edge e , treating each segment as a perfect cylinder with constant radius – see Figure 3.8.

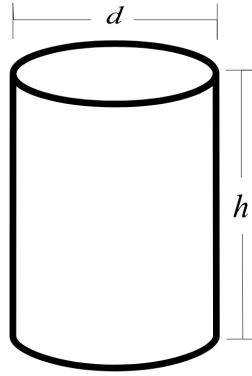


Figure 3.8: A cylinder with diameter d and height h . Image from [Wik06]

This simpler representation reduces the number of radius values to N for a rod with N edges. It also simplifies the volume preservation constraint, as each segment only depends on a single radius value for its volume calculation. However, this may lead to less smooth transitions with abrupt changes in radius at neighboring segments.

For the purposes of this thesis, the vertex-based radius representation will be adopted due to its smooth transitions and visual fidelity, despite the added complexity in constraint enforcement. This approach aligns better with the original VIPER formulation [Ang+19] and provides better accuracy for muscle fiber simulation.

Position-based dynamics framework

4

The previous Chapter 3 established the continuous formulation of volume-invariant Cosserat rods and their discretization into positions, orientations and radii. This chapter introduces **Position-Based Dynamics** (PBD) and its extended variant XPBD as the computational framework for evolving these rods over time. PBD provides a robust and efficient approach to constraint-based simulation and has become widely adopted in computer graphics and physical simulation for real-time applications.

4.1 PBD fundamentals

Position-Based Dynamics, introduced by Müller et al. [Mül+07], represents a shift from traditional force-based simulation methods. Instead of computing forces, integrating accelerations, and correcting constraint violations, PBD directly manipulates the positions of particles through constraint projection. This allows for stable and controllable simulations suited for real-time and interactive applications.

4.1.1 Simulation overview

In PBD, the objects are represented as a system of particles with each particle i having a position $\mathbf{x}_i$, velocity $\mathbf{v}_i$ and mass m_i . In practice, the inverse mass $w_i = \frac{1}{m_i}$ is used instead as it allows for particles with infinite mass (fixed particles) to be represented by $w_i = 0$. The simulation iterates in discrete time steps Δt through the following phases:

1. **Prediction:** The particle positions are predicted based on their current velocities and external forces (e.g., gravity):

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \frac{\mathbf{f}_{ext}}{m_i}, \quad (4.1)$$

$$\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i, \quad (4.2)$$

where $\mathbf{p}_i$ is the predicted position and $\mathbf{f}_{ext}$ represents external forces.

2. **Constraint projection:** The predicted positions $\mathbf{p}_i$ are iteratively adjusted to satisfy a set of constraints. This is the core of the PBD approach and is detailed in the following subsections.
3. **Velocity update:** After constraint projection converges, the velocities are updated according to the position changes:

$$\mathbf{v}_i \leftarrow \frac{\mathbf{p}_i - \mathbf{x}_i}{\Delta t}, \quad (4.3)$$

$$\mathbf{x}_i \leftarrow \mathbf{p}_i. \quad (4.4)$$

The complete PBD algorithm is presented in Algorithm 1.

Algorithm 1: Position-Based Dynamics algorithm [BMM17]

```

1 forall particles  $i$  do
2   | initialize  $\mathbf{x}_i, \mathbf{v}_i, w_i = 1/m_i$ 
3 end
4 loop
5   | forall particles  $i$  do  $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i f_{ext}$ 
6   | forall particles  $i$  do  $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$ 
7   | forall particles  $i$  do GenerateCollisionConstraints ( $\mathbf{x}_i \rightarrow \mathbf{p}_i$ )
8   | loop solverIterations times
9   |   | ProjectConstraints( $C_1, \dots, C_{M+M_{coll}}, \mathbf{p}_1, \dots, \mathbf{p}_N$ )
10  | end
11  | forall particles  $i$  do
12  |   |  $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i) / \Delta t$ 
13  |   |  $\mathbf{x}_i \leftarrow \mathbf{p}_i$ 
14  | end
15  | VelocityUpdate( $\mathbf{v}_1, \dots, \mathbf{v}_N$ )
16 end

```

The algorithm starts with the initialization of particle positions, velocities and inverse masses (lines 1–3). Next, velocities are updated by applying external forces and positions are predicted using a semi-implicit Euler integration (lines 5–6). Collision constraints are then generated based on the predicted positions (line 7). The core of the algorithm is the constraint projection phase (lines 8–10), where the ProjectConstraints function is called repeatedly for a fixed number of solver iterations. Within each iteration, all constraints are processed sequentially. Finally, velocities are computed from the position changes (line 12) and positions are updated (line 13). The VelocityUpdate function (line 14) applies additional velocity changes such as damping or friction.

4.1.2 Constraint formulation

Within the context of PBD, constraints are expressed as scalar functions $C(\mathbf{x}_1, \dots, \mathbf{x}_n)$ that evaluate to zero when satisfied. For example, a distance constraint between two particles x_i and x_j that maintain a fixed distance d would be formulated as follows:

$$C(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| - d = 0. \quad (4.5)$$

Each constraint may involve more than one particle and can represent various physical properties such as distance, angle, volume, etc. The goal of the projection phase is to find positional corrections $\Delta\mathbf{x}_i$ so that $C(\mathbf{x}_1 + \Delta\mathbf{x}_1, \dots, \mathbf{x}_n + \Delta\mathbf{x}_n) = 0$.

4.1.3 Constraint projection

The position corrections are computed by linearizing the constraint function around the current positions. For a constraint C involving particles with positions $\mathbf{x}_1, \dots, \mathbf{x}_n$, the correction for particle i is:

$$\Delta\mathbf{x}_i = -s w_i \nabla_{\mathbf{x}_i} C, \quad (4.6)$$

where $\nabla_{\mathbf{x}_i} C$ is the gradient of C with respect to $\mathbf{x}_i$, and the scaling factor s is given by:

$$s = \frac{C}{\sum_j w_j \|\nabla_{\mathbf{x}_j} C\|^2}. \quad (4.7)$$

This formulation ensures that the corrections are proportional to the particle masses (i.e., particles with less mass move more) and are aligned with the constraint gradient – the direction of the steepest descent.

4.1.4 Iterative solving

Multiple constraint types cannot be satisfied simultaneously within a single constraint projection. Therefore, PBD employs a Gauss-Seidel-type iterative approach – constraint types are solved sequentially, with each constraint projection immediately updating particle positions before the next constraint type is processed [Mül+07]. This process is repeated for n_{iter} iterations.

The order in which constraints are solved can affect the rate of convergence. It is crucial to keep the order of constraints (especially for over-constrained situations) to avoid oscillations [Mül+07].

4.1.5 Stiffness control and its limitations

PBD introduces a stiffness parameter $k \in [0, 1]$ that defines the strength of a constraint and scales the positional corrections as follows:

$$\Delta\mathbf{x}_i = -k s w_i \nabla_{\mathbf{x}_i} C. \quad (4.8)$$

A stiffness of $k = 1$ enforces the constraints fully, while lower values produce softer behavior allowing for constraints to be violated to a certain degree. However, this approach has a fundamental limitation – the effective stiffness depends not only on k but also on the number of solver iterations n_{iter} and the time step Δt . Both increasing the number of iterations and decreasing the time step make materials stiffer, even when a constant stiffness k is used.

Müller et al. [Mül+07] have partially addressed this issue by exponentially scaling the stiffness parameter based on the number of iterations:

$$k' = 1 - (1 - k)^{1/n_{iter}}. \quad (4.9)$$

While this leads to consistent behavior across different iteration counts, it does not solve the dependency on the time step Δt – smaller time steps still lead to stiffer materials. This limitation motivated the development of the **Extended Position-Based Dynamics** algorithm, which is the focus of the next section.

4.2 XPBD extension

Extended Position-Based Dynamics (XPBD), introduced by Macklin et al. [MMC16], addresses the stiffness control limitations of PBD by reformulating constraints into well-defined concepts of elastic potential energy exploiting a Lagrange multiplier approach. This allows for precise control over material stiffness independent of the number of solver iterations or the time step size.

4.2.1 Compliance-based formulation

Instead of using a dimensionless stiffness parameter k , XPBD introduces a **compliance** parameter α with units of inverse stiffness m/N . The compliance represents how much a constraint is allowed to deviate under unit force.

The time-scaled compliance $\tilde{\alpha}$ is calculated as follows [MMC16; Sai+24]:

$$\tilde{\alpha} = \frac{\alpha}{\Delta t^2}. \quad (4.10)$$

This scaling ensures that the effective stiffness of the constraint remains constant regardless of the time step. A compliance of $\alpha = 0$ corresponds to a non-violable constraint and is identical to PBD with $k = 1$ while higher values correspond to softer behavior.

4.2.2 Lagrange multiplier formulation

XPBD introduces a Lagrange multiplier λ for each constraint, which represents the accumulated constraint force over the duration of the simulation. The constraint

projection step is reformulated to solve for both the position corrections Δx (Equation 4.12) and the change in the Lagrange multiplier $\Delta\lambda$ (Equation 4.11) simultaneously. For an inverse mass matrix $\mathbf{M}^{-1}$, constraint C_j and position x_i the changes are computed as follows:

$$\Delta\lambda_j \leftarrow \frac{-C_j(x_i) - \tilde{\alpha}_j\lambda_{ij}}{\nabla C_j M^{-1} \nabla C_j^T + \tilde{\alpha}_j}, \quad (4.11)$$

$$\Delta x \leftarrow M^{-1} \nabla C(x_i)^T \Delta\lambda. \quad (4.12)$$

The position and Lagrange multiplier are then updated:

$$\lambda_{i+1} \leftarrow \lambda_i + \Delta\lambda, \quad (4.13)$$

$$x_{i+1} \leftarrow x_i + \Delta x. \quad (4.14)$$

The Lagrange multiplier λ is initialized to 0 at the beginning of each time step and accumulates across solver iterations. This accumulation is crucial for achieving independence from the time step size and iteration count.

4.2.3 XPBD simulation loop

The overall algorithm structure remains similar to PBD (Algorithm 1), with the key difference being the inner constraint projection step which now uses the XPBD formulation – the inner loop is described in Algorithm 2.

Algorithm 2: Inner loop of the XPBD algorithm [MMC16; Sai+24]

```

1 predict  $\tilde{x} \leftarrow x_n + \Delta t v_n + \Delta t^2 M^{-1} f_{ext}(x_n)$ 
2 initialize  $x_0 \leftarrow \tilde{x}$ 
3 initialize  $\lambda_0 \leftarrow 0$ 
4 while  $i < solverIterations$  do
5   forall constraints do
6     compute  $C(x)$  and  $\nabla C(x)$ 
7     compute  $\Delta\lambda$  using Equation 4.11
8     compute  $\Delta x$  using Equation 4.12
9      $\lambda_{i+1} \leftarrow \lambda_i + \Delta\lambda$ 
10     $x_{i+1} \leftarrow x_i + \Delta x$ 
11  end
12   $i \leftarrow i + 1$ 
13 end
14 update velocities  $v_{n+1} \leftarrow (x_{n+1} - x_n)/\Delta t$ 
```

The simulation begins by predicting positions using current velocities and external forces (line 1). Then the predicted positions x_0 and Lagrange multipliers λ_0 are

initialized (lines 2–3), which is essential for the compliance-based approach of XPBD to work. The core of the algorithm (lines 5–11) iterates over all constraints, calculates the changes to the Lagrange multipliers and positions using Equations 4.11 and 4.12 respectively, and finally updates them. Once all solver iterations are complete, the velocities are updated based on the final positions (line 14).

4.3 Comparison with standard PBD

The key differences between XPBD and the original PBD are:

- **Stiffness specification:** PBD uses a dimensionless parameter $k \in [0, 1]$ dependent on simulation parameters, while XPBD uses a compliance parameter α with physical units corresponding to material properties.
- **Iteration count and time step independence:** XPBD's time-scaled compliance parameter allows for consistent stiffness regardless of the number of iterations or the time step size.
- **Lagrange multipliers:** XPBD introduces λ to accumulate constraint forces over time, enabling force estimation and consistent behavior. Setting $\alpha = 0$ recovers the original PBD behavior with $k = 1$.

For muscle simulation, XPBD is the preferred choice over PBD due to its consistent behavior and physical interpretability. It also allows for the direct tuning of material properties independent of the simulation parameters.

Summary

This chapter presented the Position-Based Dynamics framework and its extended variant XPBD as the basis for the muscle deformation system developed in this thesis. The key concepts covered include:

- The PBD algorithm with its intuitive ***predict-project-update*** structure and iterative solving using Gauss-Seidel iteration
- Constraint formulation as scalar functions and the gradient-based projection for positional corrections
- The limitation of stiffness control in PBD and its dependency on iteration count and time step size
- XPBD's compliance-based formulation with Lagrange multipliers that allows for stiffness control independent of simulation parameters

With the framework established, the following chapter will focus on related works in muscle deformation, providing context for the proposed solution. The specific constraints for rod simulation (i.e., volume preservation, bending, twisting, etc.) will be discussed in greater detail in Chapter 6 as part of the proposed method.

This thesis builds upon prior research spanning musculoskeletal modeling, muscle simulation, fiber generation algorithms, and rod-based physics in computer graphics. Understanding these foundations is essential for appreciating both the capabilities and limitations of existing approaches, as well as the contributions of this work.

This chapter reviews the relevant literature in four main areas. First, the Muscle Wrapping 2.0 framework is introduced as the software ecosystem within which this thesis operates. Second, surface-based muscle deformation methods are put under examination, focusing on the Position-Based Dynamics (PBD) and Extended Position Based Dynamics (XPBD) implementations that precede this work. Third, internal fiber generation methods are discussed, including template-based and point-sampling approaches for decomposing muscles into discrete fibers. Fourth, rod-based simulation techniques from computer graphics are reviewed, with an emphasis on Cosserat rod implementations and the Volume Invariant Position-based Elastic Rods (VIPER) method. Finally, a gap analysis identifies the limitations of current approaches and motivates the proposed solution.

5.1 Muscle Wrapping 2.0

Muscle Wrapping 2.0 is a collaborative research project co-developed by the Department of Computer Science and Engineering at the University of West Bohemia and the Department of Civil and Environmental Engineering at Imperial College London [MK20]. The project addresses two fundamental challenges in musculoskeletal modeling: (a) wrapping muscles, represented as triangular surface meshes around moving bones during skeletal motion, and (b) automatically decomposing these muscles into mechanical lines of action for further biomechanical analysis.

5.1.1 Project objectives and scope

The primary motivation behind Muscle Wrapping 2.0 stems from the limitations in conventional musculoskeletal models, where muscles are typically represented

by simple polylines connecting origin and insertion points [KK14]. While computationally efficient, this simplification sacrifices anatomical accuracy and fails to capture the complex geometry of muscles and their interaction with bones during movement. Kohout and Kukačka [KK14] demonstrated that this oversimplification leads to loss of musculotendon mechanics and less accurate force predictions.

Muscle Wrapping 2.0 overcomes these limitations by working with detailed 3D surface meshes of muscles, obtained from medical imaging data such as MRI. The framework computes how these meshes deform and wrap (hence the name) around the underlying bone geometry during motion. With the muscle geometry established for a given pose, the system can generate an arbitrary number of internal muscle fibers¹ for visualization or biomechanical computation.

The framework was designed with a wide range of applicability in mind, including:

- Estimation of contact forces on lower limb joints during various activities
- Simulation of musculotendon contraction mechanisms
- Surgical planning and decision support for orthopedic procedures
- Educational visualization of muscle anatomy and function

5.1.2 System architecture

The software architecture of Muscle Wrapping 2.0, illustrated in Figure 5.1, consists of two main applications `OsimMuscleGenerationTool` and `AttachmentEstimation`, supported by a shared core library and several dependencies.

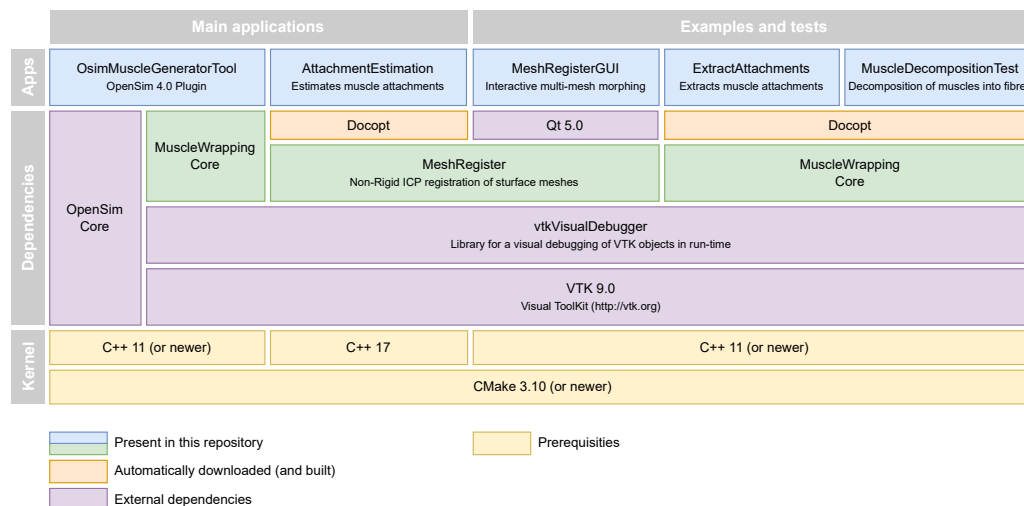


Figure 5.1: Architecture of Muscle Wrapping 2.0. Image from [Hoa23]

¹ *Internal muscle fiber* used in this context refers to the muscle-tendon unit, see Subsection 2.1.4

OsimMuscleGeneratorTool. The primary application is implemented as plugin for OpenSim, an open source platform for biomechanical modeling and simulation. This tool handles the core functionality of the framework: muscle geometry processing and deformation computation during skeletal motion. The deformation can be performed using two algorithms: the PBD-based method by Červenka [Čer19] and the XPBD-based method by Havlíček [Hav24] – discussed in detail in Section 5.2.

AttachmentEstimation. The secondary application provides tools for estimating muscle attachment areas on bone structure. Accurate specification of origin and insertion areas is crucial for realistic muscle behavior, as these attachment areas define the boundaries for both deformation and fiber generation.

Core libraries. The MuscleWrapping Core library contains shared functionality used by both applications, including data structures and algorithms. The framework also relies on several external libraries:

- OpenSim Core – for integration with biomechanical simulation workflows
- VTK (Visualization Toolkit) – for 3D mesh operations, visualization and geometric processing as well as file I/O
- Eigen – for linear algebra and matrix operations
- Discregrid – for signed distance field (SDF) computation used in collision detection

The entire project is built using CMake and requires C++20 or newer, showcasing modern software engineering practices.

5.2 Surface-based muscle deformation

Prior to this thesis, muscle deformation in the Muscle Wrapping 2.0 framework was computed using methods that operate solely on the muscle surface mesh. This section examines the two main approaches: the original PBD-based method by Červenka [Čer19] and its XPBD-based successor by Havlíček [Hav24].

5.2.1 PBD-based deformation

Červenka [Čer19] developed the first Position Based Dynamics implementation for muscle deformation within Muscle Wrapping 2.0. The method treats the muscle surface mesh as a deformable body, where individual mesh vertices serve as PBD

particles and the edges define distance constraints that resist stretching. See Figure 5.2 for an example of deformation for the gluteus medius during hip flexion from 0° to 90° and back.

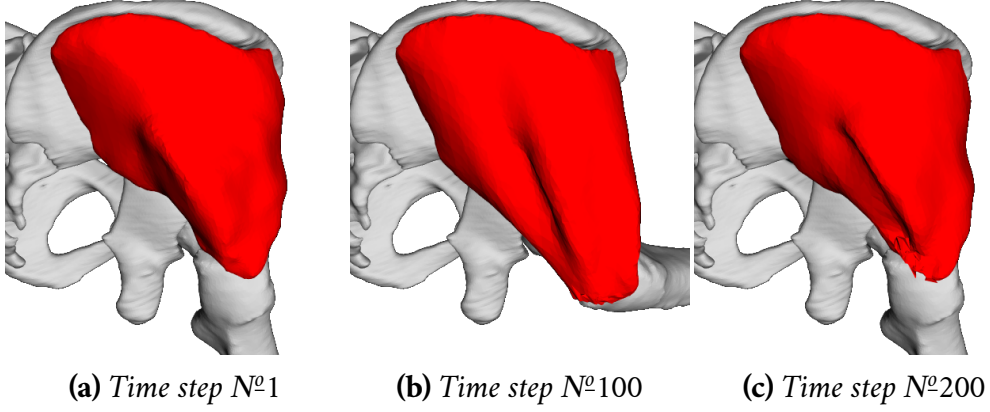


Figure 5.2: *Gluteus medius deformation over 1, 100, and 200 simulation steps using Červenka’s PBD method. Please note the form preservation issues at the 200-th simulation step surrounding the attachment area. Images from [Čer19]*

Constraint system. The deformation is governed by several types of constraints: distance constraints maintain edge length between neighboring vertices (stretch resistance), dihedral angle constraints preserve angle between adjacent triangles (bend resistance), volume constraints attempt to maintain overall muscle volume during deformation, and attachment constraints fix vertices to their respective attachment areas on bones.

Anisotropic behavior. Červenka notably contributed with the incorporation of muscle anisotropy into the constraint system. Real muscle tissue exhibits direction-dependent mechanical properties, being stiffer along fiber directions. To capture this behavior, the distance constraint stiffness is based on the alignment between edge direction and the local fiber orientation. Havlíček [Hav24] described this stiffness modulation for edge e as:

$$k_e = 1 - \mathbf{u}_e \cdot \mathbf{v}_e,$$

where $\mathbf{u}_e$ is the normalized edge direction and $\mathbf{v}_e$ is the tangent direction of the nearest surface fiber. When these vectors are parallel (i.e., edge aligned with fiber), the stiffness k_e approaches 0, when perpendicular, stiffness remains at 1.

Muscle-bone collision handling. Muscle to bone collision detection and response was solved using a **signed distance field** (SDF) generated via the Discregrid library. The SDF works by discretizing the space around each bone into a voxel grid, where

each voxel stores distance and gradient information to the nearest bone surface. When a muscle vertex penetrates a bone (negative distance), it is projected back to the bone surface along the gradient direction. To address the **tunneling problem** – where vertices pass completely through thin bone structures in a single time step – Červenka implemented detection by monitoring gradient direction changes between consecutive frames.

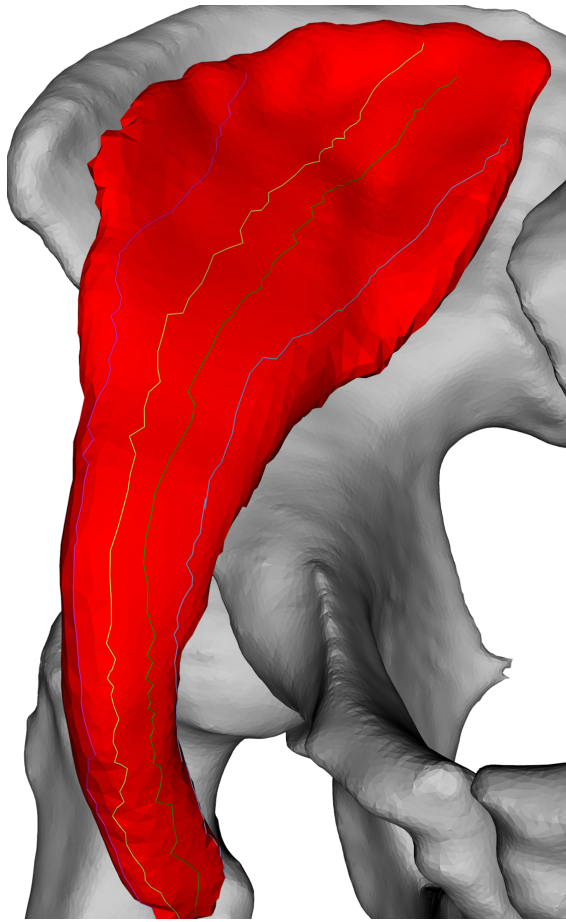
5.2.2 XPBD-based deformation

Havlíček [Hav24] extended Červenka’s PBD approach by implementing the Extended Position Based Dynamics (XPBD) algorithm [MMC16] and addressed several limitations of the original method while adding new capabilities for physiologically-driven muscle behavior.

Time step independence. The primary motivation for adopting XPBD was to eliminate the time step dependence in PBD. As discussed in Chapter 4, PBD constraints become increasingly stiff with the number of solver iterations and decreasing time step size, which made it difficult to tune material parameters independently from the simulation parameters. By reformulating the constraints using compliance parameters and Lagrange multipliers, Havlíček achieved consistent muscle stiffness regardless of both time step and iteration count.

Muscle-muscle collision handling. While muscle-bone collision handling was inherited from Červenka’s PBD implementation, Havlíček introduced a novel muscle to muscle collision detection and resolving mechanism. The inter-muscular collision system uses **virtual edges** connecting vertices of neighboring muscles. When the distance between connected vertices falls below a certain proximity threshold, the collision is resolved by pushing the vertices apart. This method employs a **Zig-Zag** algorithm to iteratively refine vertex positions, allowing for sequential resolution of multiple collisions.

Activation-driven contraction. A significant advancement in Havlíček’s work was the integration of activation-driven muscle contractions. Using activation data obtained from OpenSim, the method contracts the muscle surface along fiber directions proportional to the activation level. This is achieved by extracting surface fiber paths from internal fiber decomposition and modifying distance constraints along these paths to enforce shortening during activation. An illustration of detailed surface fibers is shown in Figure 5.3.



The surface fiber extraction works as follows:

1. internal muscle fibers are generated using the Kukačka method [KK14]
2. each vertex of the generated fiber is projected to the nearest point on the muscle surface
3. the projected points are connected to form detailed surface fiber paths
4. during simulation, edge lengths along these surface paths are modified based on the activation level

Figure 5.3: *Perpendicularly running detailed surface fibers on the iliacus muscle*

Activation-driven contraction requires sufficient solver iterations for constraints to propagate through the surface mesh. While Červenka’s PBD implementation ran with only 3 solver iterations per time step, this was insufficient for proper constraint propagation. Havlíček’s XPBD implementation uses approximately 70 iterations per time step, enabling physically plausible deformation behavior. The framework was extended with 16 additional muscles that were validated under the XPBD simulation with activation support.

5.2.3 Limitations of surface-based approaches

Despite the advancements made by Červenka and Havlíček, surface-based deformation methods have fundamental limitations that motivate the approach proposed in this thesis.

Neglect of internal structure. The most significant limitation is that surface-based methods operate solely on the muscle boundary, completely ignoring the internal

fiber architecture. Real muscles are composed of thousands of muscle fibers organized in specific arrangements that affect their mechanical behavior – see Section 2.1. The collective contraction and deformation of these internal fibers drives the overall changes in the muscle. By working only with the surface, current methods cannot capture this behavior.

Indirect volume preservation. While both the PBD and XPBD implementations include volume constraints, these are applied only to the surface mesh rather than locally within the muscle tissue. This can lead to unrealistic deformations where the surface maintains global muscle volume but fails to capture the local volume redistribution that occurs during fiber contraction.

Surface fiber approximation. The activation-driven contraction in Havlíček’s method relies on surface projections of internal fibers, which is an approximation of true muscle fiber behavior. The actual shortening occurs along internal fibers throughout the muscle volume, not just on the surface. This approximation can lead to inaccuracies especially for deep fibers that have significantly different orientations than their surface projections.

Parameter sensitivity and stability. Surface-based methods exhibit sensitivity to constraint parameters that can lead to instabilities in extreme configurations. Havlíček noted that although overly stiff constraints prevented some deformation artifacts, they could cause *muscle explosions* near attachment areas where degenerated triangles form and penetrate bones. The surface fiber extraction method can also produce unnatural fiber paths on thin muscles like the psoas, which leads to over-constriction during contraction. These issues highlight a fundamental challenge for surface-based approaches, as there is no universal parameter configuration that works reliably across all muscles and poses, often requiring manual tuning.

5.3 Internal fiber generation methods

Representing muscle internal structure requires methods to decompose the muscle model into discrete fibers. This section reviews the two main approaches used within the Muscle Wrapping 2.0 framework and related literature. A quick overview is presented in Table 5.1.

5.3.1 Template-based decomposition

Kohout and Kukačka [KK14] introduced a novel template-based decomposition method, called *Kukačka* which combines harmonic field computation with prede-

finer fiber templates – see Figure 5.4. This is the first out of the two decomposition methods of Muscle Wrapping 2.0.

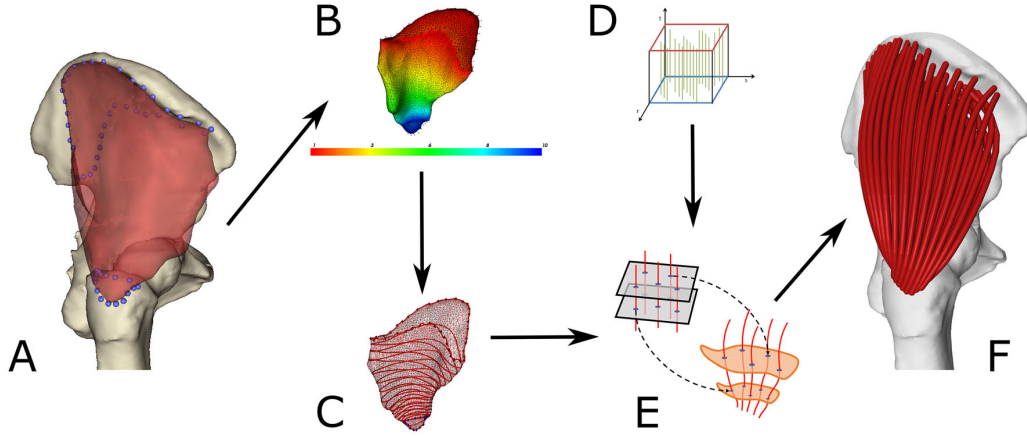


Figure 5.4: Kukačka method for muscle decomposition. *A* – program input, *B* – harmonic scalar field of the muscle surface, *C* – extraction of iso-lines, *D* – fiber template selection, *E* – merge the sliced muscle and template, *F* – program output. Image from [MK20]

The method operates in two parallel processes. The first process computes a harmonic scalar field over the muscle surface mesh, with boundaries at the origin and insertion attachment areas. Iso-lines of this field are then extracted to slice the muscle into N cross-sectional contours, where $N - 1$ corresponds to the desired number of line segments for each fiber.

The second process selects a fiber template based on the muscle’s anatomical fiber arrangement (see Subsection 2.1.2). Finally, the two processes merge, the fiber template is mapped slice by slice into the interior of the respective muscle contours by exploiting *mean value coordinates* (MVC).

5.3.2 Viperization

Angles et al. [Ang+19] introduced *viperization*, a mesh-to-rod conversion process that generates VIPER rod bundles from meshes. However, unlike the Kukačka method, it operates on volumetric mesh representations of a muscle and not surface meshes.

The viperization process, illustrated in Figure 5.5, begins by computing a harmonic function over the muscle volume given the source and sink constraints (note the difference from Kukačka which computes it over the surface). Then, K iso-levels are extracted and for each, a *restricted centroidal Voronoi diagram* (restricted CVD) is executed, placing M elements of the same radii on each iso-level. Finally, a combinatorial optimization step connects the elements across the iso-levels to produce minimal length curves.

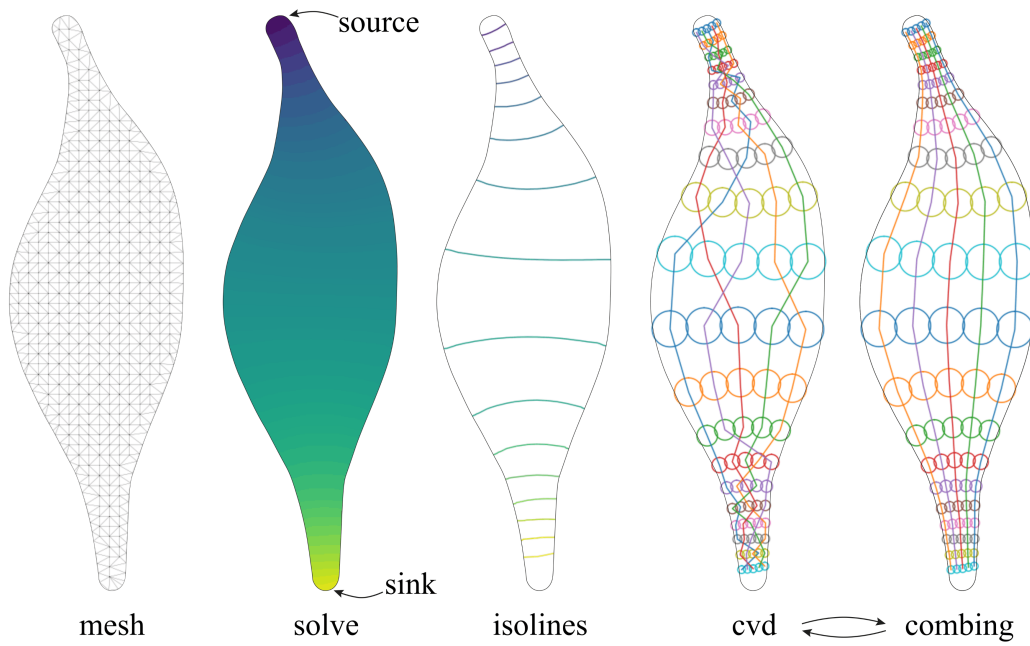


Figure 5.5: Viperization method for muscle decomposition. Image from [Ang+19]

5.3.3 Point sampling based decomposition

The second decomposition method of Muscle Wrapping 2.0 was introduced by Hoang [Hoa23] – an alternative to Kukačka that replaces template-based fiber creation with a point distribution approach, see Figure 5.6.

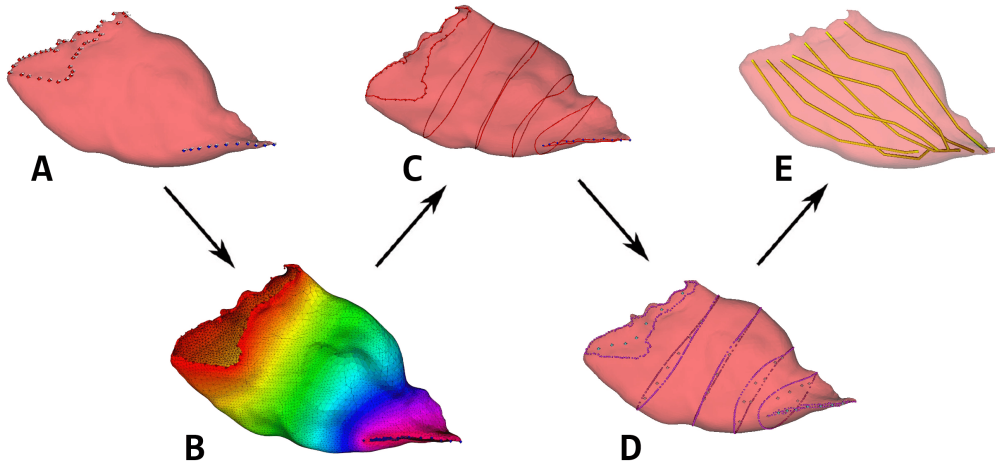


Figure 5.6: Hoang method for muscle decomposition. A – program input, B – harmonic scalar field of the muscle surface, C – extraction of iso-lines, D – point sampling on each slice, E – Munkres algorithm. Image from [Hoa23]

This method reuses the first process of the Kukačka method – the harmonic field

computation and iso-line extraction – but diverges in how fiber points are placed within each slice.

Instead of using predefined fiber templates, the Hoang method takes inspiration from VIPER’s [Ang+19] restricted CVD approach. Each contour is first projected onto a best-fit plane and transformed to 2D, where Lloyd’s algorithm computes the CVD centroids. These points are then mapped back to 3D. To connect the points across slices into fibers, the *Munkres algorithm* (Hungarian algorithm) is employed to find the optimal point-to-point correspondences that minimize overall fiber length.

Comparative evaluation showed that fibers generated by the Hoang decomposition method are comparable to Kukačka in terms of length distribution, while filling out more of the muscle volume due to the point sampling approach. The method was also rated higher for plausibility of the generated fiber paths by a general audience. Although the Hoang method has a higher computational complexity than its predecessor, it still executes under 1 second for 100 generated fibers, making it suitable for use in real-time applications.

Table 5.1: Comparison of relevant approaches for muscle decomposition

Method	Muscle model	Iso-level/line extraction	Fiber creation
Kukačka	surface mesh	harmonic solve	fiber template
VIPER	volume mesh	harmonic solve	restricted CVD
Hoang	surface mesh	harmonic solve	restricted CVD

5.4 Rod-based deformation in computer graphics

Rod-based simulation has been an active research area in computer graphics for decades, with applications ranging from hair and fur simulation to cables, ropes and biological structures. This section reviews PBD-compatible rod implementations and the VIPER extension, building on the foundations presented in Chapter 3.

5.4.1 Cosserat rods in PBD

Integrating Cosserat rod theory into the PBD framework requires both positions and orientations to be represented, which poses a challenge as classical PBD operates solely on particles. Two approaches have been proposed to address this.

Ghost points. Umetani et al. [USS14] introduced auxiliary *ghost points* positioned perpendicular to each rod segment to represent the material frame orientation.

Constraints force these ghost points to maintain their relative positions during simulation, effectively tracking the frame's rotation. Although elegant in its simplicity and compatibility with standard PBD solvers, this approach increases the number of particles and constraints, which can lead to performance issues and could suffer from numerical instability in high-torsion scenarios.

Quaternion-based frames. Kugelstadt and Schömer [KS16] extended the PBD framework to directly support orientation constraints using unit quaternions. Instead of encoding the material frames through ghost points, orientations are treated as first-class simulation variables alongside positions, leading to simpler constraint formulations. This avoids trigonometric functions and the overhead of additional particles, achieving better performance compared to the ghost points approach. This quaternion-based approach forms the basis for rod simulation in many modern PBD implementations, and this thesis is no exception.

5.4.2 Volume Invariant Position-based Elastic Rods

As introduced in Chapter 3, the VIPER method [Ang+19] extends classical Cosserat rods by an additional degree of freedom, the radius $r(s)$, enforcing constant volume during deformation.

Muscle modeling with rod bundles. VIPER demonstrates that muscles can be efficiently represented as bundles of volume-invariant rods. The physics simulation operates on this rod subspace, and the results are transferred to the muscle surface mesh via linear blend skinning. This approach achieves results similar to expensive volumetric FEM simulations, while being computationally more efficient and unlike surface-based methods, can capture internal fiber behavior with real-time performance (up to 26000 DOFs at 140 Hz on a GPU [Ang+19]).

Collision handling. The rod representation allows for efficient collision detection through the representation of every volumetric object as a tapered capsule (pill) for each rod segment. Following a GPU-friendly approach by Green [Gre10], each pill is approximated by its bounding sphere and collisions are detected with the help of a uniform grid structure.

Limitations for biomechanical applications. Although VIPER demonstrates muscle simulation as an application, it was designed primarily for real-time graphics and character modeling instead of biomechanical analysis. The method does not incorporate physiological muscle activation models and the implementation is not integrated into established biomechanical frameworks like OpenSim.

5.4.3 Recent advances

Recently, Hsu et al. [Hsu+25] introduced **stable Cosserat rods**, a novel simulation scheme that works even under large time steps. The key insight is that the angular momentum contributes negligibly to the thin Cosserat rod dynamics, allowing for a reformulation of the original optimization problem into two interleaved alternating optimizations – one for positions and one for orientations. This decoupling allows for highly efficient parallelization while improving stability and performance without sacrificing accuracy. Comparative experiments demonstrated that stable Cosserat rods outperform prior methods like XPBD, **Vertex Block Descent** (VBD) and **Discrete Elastic Rods** (DER) by more than an order of magnitude in terms of computational efficiency.

5.5 Gap analysis and thesis contribution

The previous sections have surveyed relevant literature in surface-based deformation, fiber generation and rod-based simulation. This section summarizes the key findings to identify the research gap that this thesis aims to address.

5.5.1 Summary of existing approaches

Table 5.2 summarizes the key characteristics of existing methods relevant to muscle deformation simulation as well as the proposed approach of this thesis:

Table 5.2: Comparison of existing approaches for muscle simulation

Method	Internal fibers	Volume preservation	Activation support	OpenSim integration
Červenka (PBD)	✗	Global	✗	✓
Havlíček (XPBD)	✗	Global	✓ ^a	✓
VIPER	✓	Local	✗	✗
This thesis	✓	Local	✓ ^b	✓

^a surface fibers ^b internal fibers

The surface-based predecessors to this thesis (Červenka, Havlíček) operate within the Muscle Wrapping 2.0 framework and benefit from OpenSim integration, though they lack internal fiber representation and volume preservation happens only at the global scale. In contrast, VIPER employs internal fibers and local volume preservation but lacks activation support and OpenSim integration.

5.5.2 Identified gap

No existing method combines all of the following:

1. internal fiber representation that captures muscle architecture
2. local volume preservation that is enforced at the fiber level
3. physiologically-driven activation that contracts fibers
4. integration with Muscle Wrapping 2.0 and OpenSim

This identified gap represents a significant opportunity. By bringing the VIPER-style volume-invariant rods into the activation-driven XPBD framework of Muscle Wrapping 2.0, it becomes possible to simulate muscles with both internal fiber representation and practical usability.

5.5.3 Thesis contribution

This thesis proposes to bridge this gap by developing a method that represents muscles as volume-invariant Cosserat rods within the Muscle Wrapping 2.0 framework. The key contributions are:

1. **Rod-based muscle fiber representation** – Muscles are decomposed into VIPER-style rods that capture internal fiber architecture, replacing surface-only deformation with volumetric fiber-based simulation
2. **Volume preservation at the fiber level** – Volume preserving constraints are applied to each rod, ensuring physiologically plausible volume changes during deformation.
3. **Activation-driven deformation** – Muscle activation signals from OpenSim drive rod contraction, with the deformation behavior emerging naturally from within and outwards to the muscle surface mesh.
4. **Framework integration** – The implementation fully integrates into Muscle Wrapping 2.0, maintaining compatibility with existing workflows and enabling direct comparison with previous methods.

The following chapter presents the proposed method in detail.

Proposed method

6

This chapter presents the core contribution of this thesis, a method for simulating muscle deformation using volume-invariant Cosserat rods within the XPBD implementation of the Muscle Wrapping 2.0 framework. Building upon the theoretical foundations established in Chapter 3 and the gap analysis in Chapter 5, the proposed approach represents the muscles as bundles of VIPER-style rods that capture internal fiber architecture while enforcing local volume preservation.

The chapter is structured as follows. Section 6.1 describes the conversion of muscle fibers into discrete Cosserat rod representations suitable for simulation. Section 6.2 details the XPBD constraint formulations for rod mechanics as well as volume preservation. Section 6.3 discusses the collision detection and response mechanisms for rod-bone and rod-rod interactions. Section 6.4 explains how physiological activation signals drive muscle contraction. Finally, Section 6.5 provides a system overview, describing how these components integrate into the Muscle Wrapping 2.0 framework.

6.1 Rod-based muscle representation

This section describes how muscle-tendon units are represented as collections of volume-invariant Cosserat rods. The representation bridges the gap between the fiber decomposition methods used in Muscle Wrapping 2.0 and the discrete rod model required for simulation.

6.1.1 From fibers to rods

The Muscle Wrapping 2.0 framework provides muscle fibers as polylines obtained through muscle decomposition methods like Kukačka [KK14] or Hoang [Hoa23]. These *fibers* represent the directional organization of the muscle-tendon unit from origin to insertion, abstracting the underlying fascicle structure into computationally tractable curves.

The conversion process enriches each fiber with the necessary rod attributes:

Centerline initialization. The fiber polyline vertices $\{\mathbf{f}_0, \mathbf{f}_1, \dots, \mathbf{f}_n\}$ directly map to the rod centerline points $\{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n\}$. The edge lengths in this initial configuration define the reference lengths $\hat{l}_e = \|\mathbf{p}_e - \mathbf{p}_{e-1}\|$ used by the stretch-shear constraint.

Reference frame initialization. For each edge e , an initial reference frame must be established. The tangent direction $\mathbf{d}_3$ is determined by the edge direction:

$$\mathbf{d}_3^e = \frac{\mathbf{p}_e - \mathbf{p}_{e-1}}{\|\mathbf{p}_e - \mathbf{p}_{e-1}\|}.$$

The remaining directors $\mathbf{d}_1$ and $\mathbf{d}_2$ are constructed to form an orthonormal basis. A robust approach selects an arbitrary vector $\mathbf{u}$ not parallel to $\mathbf{d}_3$ and computes:

$$\begin{aligned}\mathbf{d}_1 &= \frac{\mathbf{u} \times \mathbf{d}_3}{\|\mathbf{u} \times \mathbf{d}_3\|}, \\ \mathbf{d}_2 &= \mathbf{d}_3 \times \mathbf{d}_1.\end{aligned}$$

The resulting frame is converted to a unit quaternion $\mathbf{q}_e$ representing the rotation from the world frame to the local frame, identically to the approach of Kugelstadt and Schömer [KS16].

Radius initialization. Each vertex of the rod centerline is assigned an initial radius r_i based on the muscle geometry. The original VIPER approach [Ang+19], assigns the same radii to all points within each iso-level, though other strategies should be explored:

1. **Uniform radius per slice (VIPER)** – All K points on slice i share radius r_i , which is calculated from the slice area $r_i \approx \sqrt{\frac{A_i}{K \cdot \pi \cdot \eta}}$ where $\eta \approx 0.9$ is a packing efficiency factor. This ensures smooth radius transitions along rods but may inefficiently fill irregular contours, leaving unused space near concavities and over-constraining points near the boundary.
2. **Min distance-based variable radius** – Each point p on slice i is assigned radius r_p equal to half the distance to the nearest neighbor or muscle boundary. This approach adapts well to irregular shapes but can lead to abrupt radius changes along individual rods.
3. **Smoothed variable radius** – Expands the previous neighbor-based approach by applying a 1D smoothing filter (e.g., Gaussian or moving average) along each rod. This balances adaptability to shape with gradual radius transitions.

This thesis adopts the **min distance-based variable radius** approach. For each point, the initial radius is computed as half the distance to the nearest neighbor

within the same slice, optionally clamped by the distance to the muscle contour boundary with a small tolerance factor to avoid excessive shrinking and allow for slight boundary overflow. The radii are then smoothed along each rod using a Gaussian filter with a small kernel (around 3–5 points) to eliminate abrupt transitions while preserving the overall muscle shape.

6.1.2 Discrete rod model

The discrete rod model, introduced in Section 3.3, represents a continuous Cosserat rod as a chain of vertices connected by edges, with orientations defined at edges or vertices. This section provides a summary of the discretization choices made for the proposed method.

Vertex representation. Each vertex i of a rod stores:

- **Position:** $\mathbf{p}_i \in \mathbb{R}^3$
- **Radius:** $r_i \in \mathbb{R}^+$
- **Velocity:** $\mathbf{v}_i \in \mathbb{R}^3$
- **Mass:** $m_i \in \mathbb{R}^+$

Edge representation. Each edge e connecting vertices $i - 1$ and i stores:

- **Reference length:** $\hat{l}_e \in \mathbb{R}^+$
- **Reference Darboux vector:** $\hat{\Omega}_e \in \mathbb{R}^3$
- **Angular velocity:** $\omega_e \in \mathbb{R}^3$
- **Orientation quaternion:** $\mathbf{q}_e \in \mathbb{H}$ (unit quaternion)

This edge-based orientation scheme, illustrated in Figure 3.5, follows the approach of Kugelstadt and Schömer [KS16] rather than the ghost point approach of Umetani et al. [USS14]. As discussed in Subsection 3.3.2, quaternions provide computational efficiency, avoid gimbal lock, and integrate naturally into the constraint-based PBD framework.

Segment geometry. Each edge e and its endpoint vertices $i - 1$ and i form a truncated cone (frustum) with radii r_{i-1} and r_i , and a length l_e . The volume of such a segment is calculated as follows:

$$V = \frac{1}{3}\pi l_e (r_{i-1}^2 + r_{i-1}r_i + r_i^2),$$

where $l_e = \|\mathbf{p}_i - \mathbf{p}_{i-1}\|$ is the current edge length.

6.1.3 Rod bundle configuration

A single rod captures the behavior of an individual muscle fiber path within the muscle-tendon unit (MTU), however realistic muscle simulation requires representing the collective behavior of multiple fibers paths organized into a coherent structure. Following VIPER [Ang+19], muscles are modeled as a bundle of rods that interact and deform together while maintaining a relative spatial arrangement.

Bundle organization. Rods within a bundle share common attachment points at the muscle's origin and insertion areas. The fiber generation methods of Muscle Wrapping 2.0, Kukačka and Hoang, naturally produce fibers that span from origin to insertion, providing the initial bundle configuration.

Cross-sectional coherence. In the original VIPER [Ang+19], a shape matching constraint encourages rods within a cross-section to undergo similar transformations. However, with fixed attachment areas at both ends, the stretch-shear and bend-twist constraints naturally maintain bundle coherence without requiring explicit cross-sectional coupling.

Practical considerations. The number of rods per bundle and the number of segments per rod are user-configurable parameters that trade off simulation fidelity against computational cost. Typical configurations for muscles in Muscle Wrapping 2.0 use 50–100 rods with 10–15 segments each.

6.2 XPBD constraint formulation

This section presents the XPBD constraint formulations that govern rod mechanics. Building upon the XPBD framework introduced in Section 4.2, each constraint is defined by a constraint function C , gradient ∇C , and a compliance parameter α . The constraints work together to produce physically plausible rod behavior while allowing for efficient simulation.

The general XPBD update for a constraint C_j with compliance $\tilde{\alpha}_j = \alpha_j / \Delta t^2$ and accumulated Lagrange multiplier λ_j is:

$$\Delta \lambda_j = \frac{-C_j(\mathbf{p}) - \tilde{\alpha}_j \lambda_j}{\|\nabla C_j\|^2 + \tilde{\alpha}_j}, \quad (6.1)$$

$$\Delta \mathbf{p}_i = \nabla_{\mathbf{p}_i} C_j \cdot \Delta \lambda_j. \quad (6.2)$$

6.2.1 Stretch-shear constraints

The stretch-shear constraint maintains edge lengths close to their reference values, preventing excessive elongation or compression. Following Kugelstadt and Schömer [KS16], the stretch-shear constraint for edge e is defined as:

$$\mathbf{C}_s(\mathbf{p}_{e-1}, \mathbf{p}_e, \mathbf{q}_e) = \frac{1}{\hat{l}_e}(\mathbf{p}_e - \mathbf{p}_{e-1}) - \mathbf{R}(\mathbf{q}_e)\mathbf{e}_3 = \mathbf{0}, \quad (6.3)$$

where $\mathbf{R}(\mathbf{q}_e)$ is the rotation matrix corresponding to the quaternion $\mathbf{q}_e$, and $\mathbf{e}_3 = (0, 0, 1)^T$ is the local tangent direction.

Gradients. The gradients with respect to positions and quaternion are:

$$\begin{aligned} \nabla_{\mathbf{p}_{e-1}} \mathbf{C}_s &= -\frac{1}{\hat{l}_e} \mathbf{I}_{3 \times 3}, \\ \nabla_{\mathbf{p}_e} \mathbf{C}_s &= \frac{1}{\hat{l}_e} \mathbf{I}_{3 \times 3}, \\ \nabla_{\mathbf{q}_e} \mathbf{C}_s &= 2\hat{\mathbf{Q}}(\mathbf{e}_3 \bar{\mathbf{q}}_e)_{3 \times 4}, \end{aligned}$$

where $\hat{\mathbf{Q}}(\cdot)_{3 \times 4}$ denotes the lower 3×4 block of the quaternion matrix, and $\bar{\mathbf{q}}_e$ is the quaternion conjugate.

Position and orientation corrections. Following the derivation in [KS16], the correction displacements simplify to:

$$\begin{aligned} \Delta \mathbf{p}_{e-1} &= \frac{w_{e-1} \hat{l}_e}{w_{e-1} + w_e + 4w_q \hat{l}_e^2} \mathbf{C}_s, \\ \Delta \mathbf{p}_e &= -\frac{w_e \hat{l}_e}{w_{e-1} + w_e + 4w_q \hat{l}_e^2} \mathbf{C}_s, \\ \Delta \mathbf{q}_e &= \frac{w_q \hat{l}_e^2}{w_{e-1} + w_e + 4w_q \hat{l}_e^2} \mathbf{C}_s \mathbf{q}_e \mathbf{e}_3, \end{aligned}$$

where $w_i = \frac{1}{m_i}$ are inverse masses and w_q is the inverse inertia for the quaternion.

Rest length modification for activation. A key mechanism for muscle contraction is the modification of the reference length $\hat{l}_e$ based on the activation level. When activation increases, $\hat{l}_e$ decreases, causing the stretch-shear constraint to pull the vertices closer together, simulating fiber shortening. This mechanism is detailed in Section 6.4.

6.2.2 Bend-twist constraints

The bend-twist constraint resists changes in curvature and torsion between adjacent rod segments. It couples the orientations of consecutive edges, penalizing deviations from the reference configuration.

For two neighboring edges e_i and e_{i+1} with quaternions $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$, the constraint is defined using the Darboux vector:

$$\mathbf{C}_b(\mathbf{q}_i, \mathbf{q}_{i+1}) = \text{Im}(\bar{\mathbf{q}}_i \mathbf{q}_{i+1}) - s \cdot \text{Im}(\bar{\mathbf{q}}_i^0 \mathbf{q}_{i+1}^0) = \boldsymbol{\Omega} - s \hat{\boldsymbol{\Omega}} = \mathbf{0}, \quad (6.4)$$

where $\text{Im}(\cdot)$ extracts the imaginary (vector) part of the quaternion, $\mathbf{q}_i^0$ and $\mathbf{q}_{i+1}^0$ are the rest-state quaternions, and $s = \pm 1$ selects the nearest rest pose to handle **quaternion double-cover**. The sign factor s is chosen as:

$$s = \begin{cases} +1 & \text{if } \|\boldsymbol{\Omega} - \hat{\boldsymbol{\Omega}}\|^2 < \|\boldsymbol{\Omega} + \hat{\boldsymbol{\Omega}}\|^2, \\ -1 & \text{otherwise.} \end{cases}$$

The sign factor s addresses the quaternion double-cover problem: both $\mathbf{q}$ and $-\mathbf{q}$ represent the same physical rotation. Without this correction, the constraint solver might attempt to rotate through 360° instead of taking the shorter path, causing numerical instability and visual artifacts.

Physical interpretation. The Darboux vector $\boldsymbol{\Omega} = (\Omega_1, \Omega_2, \Omega_3)$ has three components with direct physical meaning:

- Ω_1, Ω_2 : Curvature (bending) around the $\mathbf{d}_1$ and $\mathbf{d}_2$ directions
- Ω_3 : Torsion (twisting) around the tangent $\mathbf{d}_3$ direction

Gradients and corrections. The gradients with respect to the quaternions are:

$$\begin{aligned} \nabla_{\mathbf{q}_i} \mathbf{C}_b &= - \begin{pmatrix} -\text{Im}(\mathbf{q}_{i+1}) & q_{i+1,w} \mathbf{I}_{3 \times 3} - [\text{Im}(\mathbf{q}_{i+1})]_{\times} \end{pmatrix}, \\ \nabla_{\mathbf{q}_{i+1}} \mathbf{C}_b &= - \begin{pmatrix} -\text{Im}(\mathbf{q}_i) & q_{i,w} \mathbf{I}_{3 \times 3} - [\text{Im}(\mathbf{q}_i)]_{\times} \end{pmatrix}, \end{aligned}$$

where $q_{i,w}$ denotes the scalar component of the quaternion $\mathbf{q}_i$, and $[\cdot]_{\times}$ denotes the skew-symmetric cross-product matrix.

The quaternion corrections simplify to quaternion products [KS16]:

$$\begin{aligned} \Delta \mathbf{q}_i &= \frac{w_{q_i}}{w_{q_i} + w_{q_{i+1}}} \mathbf{q}_{i+1} (\boldsymbol{\Omega} - s \hat{\boldsymbol{\Omega}}), \\ \Delta \mathbf{q}_{i+1} &= - \frac{w_{q_{i+1}}}{w_{q_i} + w_{q_{i+1}}} \mathbf{q}_i (\boldsymbol{\Omega} - s \hat{\boldsymbol{\Omega}}), \end{aligned}$$

where the constraint vector is embedded as a quaternion with zero scalar part.

6.2.3 Volume preservation constraints

The volume preservation constraint is the defining feature of the VIPER rods that distinguishes them from traditional Cosserat rods. As established in Subsection 3.2.2, this constraint couples the rod's stretch to its radius, ensuring that when a rod segment stretches, it reduces its radius and vice versa to maintain constant volume.

Constraint formulation. A rod segment defined by the edge e and its endpoint vertices $i - 1$ and i is modeled as a truncated cone (frustum) with radii r_{i-1} and r_i , and length l_e . The volume preservation constraint requires that the current volume V_e equals the reference volume $\hat{V}_e$:

$$C_v = V_e - \hat{V}_e = \frac{\pi l_e}{3} (r_{i-1}^2 + r_{i-1}r_i + r_i^2) - \hat{V}_e = 0, \quad (6.5)$$

where $\hat{V}_e$ is the reference volume calculated from the rest configuration.

Physical interpretation. The constraint captures the key physical insight: when a segment shortens ($l_e < \hat{l}_e$), the radii must increase to maintain volume, producing the characteristic bulging effect. Conversely, when the segment stretches ($l_e > \hat{l}_e$), the radii must decrease, producing thinning.

Gradients. The gradients of the volume constraint with respect to positions and radii are:

$$\begin{aligned} \nabla_{\mathbf{p}_{i-1}} C_v &= -\frac{\pi}{3} (r_{i-1}^2 + r_{i-1}r_i + r_i^2) \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{l_e}, \\ \nabla_{\mathbf{p}_i} C_v &= \frac{\pi}{3} (r_{i-1}^2 + r_{i-1}r_i + r_i^2) \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{l_e}, \\ \nabla_{r_{i-1}} C_v &= \frac{\pi l_e}{3} (2r_{i-1} + r_i), \\ \nabla_{r_i} C_v &= \frac{\pi l_e}{3} (r_{i-1} + 2r_i). \end{aligned}$$

Radii-only projection. The volume constraint is designed to update only the radii, not the vertex positions. This prevents the volume constraint from competing with the stretch-shear constraint over edge lengths and this is physiologically motivated – real muscles change their cross-sectional shape to preserve volume, not their fiber length.

6.2.4 Attachment constraints

Muscles attach to bones at their origin and insertion points, and these attachments must be maintained during simulation. The attachment constraint fixes selected rod

vertices to follow bone transformations, establishing boundary conditions for the muscle deformation.

Kinematic attachment. Vertices at attachment areas are treated as kinematic particles with infinite mass ($w_i = 0$), meaning they are unaffected by constraint projections and only move according to the bone transformation:

$$\mathbf{p}_i^{\text{attached}} = \mathbf{T}_{\text{bone}}(t) \cdot \mathbf{p}_i^{\text{rest}},$$

where $\mathbf{T}_{\text{bone}}(t)$ is the bone's transformation matrix at time t .

Attachment area identification. The Muscle Wrapping 2.0 framework defines attachment areas through expert-specified landmarks that delineate the origin and insertion regions on bone surfaces [KK14; Hoa23]. For rod-based simulation, the rod endpoints are associated with these areas:

- First vertex ($\mathbf{p}_0$) of each rod attaches to the origin area
- Last vertex ($\mathbf{p}_n$) of each rod attaches to the insertion area

Since fibers by the Kukačka or Hoang methods naturally span from origin to insertion, each rod's endpoints inherently lie within the corresponding attachment areas. These endpoints inherit the transformation of the corresponding bone during simulation.

Orientation attachment. In addition to position constraints, the orientations at attachment points may also be constrained to maintain tangent alignment with the bone surface. The first and last quaternions ($\mathbf{q}_0$ and $\mathbf{q}_n$) can be partially or fully constrained based on the desired boundary behavior:

- **Free rotation:** Orientation evolves naturally from bend-twist constraints without direct attachment.
- **Fixed tangent:** Tangent direction $\mathbf{d}_3$ constrained to bone surface normal, allowing rotation around the tangent but preventing bending away from the bone.
- **Fully fixed:** Complete orientation locked to the bone's local frame, preventing any rotation at the attachment point.

For muscle simulation, free rotation at attachments typically produces the most natural deformation, allowing the fiber to adjust its entry angle as the muscle contracts.

Deformation propagation. When bones move, the attached vertices follow the bone transformation, creating a displacement that propagates through the rod via the stretch and bend-twist constraints. This wave-like propagation, also observed in the surface-based methods [Hav24], is characteristic of position-based approaches and produces visually plausible deformation patterns.

6.3 Collision handling

Collision detection and response are critical for realistic muscle simulation, preventing rods from penetrating bones and each other. This section describes the collision handling strategies for rod-bone and intra-bundle interactions, building upon the approaches established in the Muscle Wrapping 2.0 framework [Hav24].

6.3.1 Rod-bone collision

Muscles must wrap around and slide over bone surfaces during movement. The rod-bone collision system ensures that rod vertices remain outside the bone geometry while allowing natural contact behavior.

Signed distance field representation. Following the approach established by Červenka [Čer19] and continued by Havlíček [Hav24], bones are represented using signed distance fields (SDFs) generated via the Discregrid library. The SDF discretizes the space around each bone into a voxel grid, where each voxel stores the signed distance to the bone surface along with the gradient direction. For a query point $\mathbf{p}$, the SDF provides:

- **Signed distance** $d(\mathbf{p})$ – negative if inside the bone, positive if outside
- **Gradient** $\nabla d(\mathbf{p})$ – direction toward the nearest point on the bone surface

The SDF is computed once during initialization and remains constant throughout the simulation, as bones undergo only rigid transformations. Query points (rod vertices) are transformed into the bone’s local coordinate system before evaluation.

Collision detection. At each simulation step, every rod vertex $\mathbf{p}_i$ is tested against the SDFs of nearby bones. A collision is detected when the signed distance falls below a small positive threshold ϵ :

$$d(\mathbf{p}_i) < \epsilon,$$

where $\epsilon \geq 0$ provides a safety margin to prevent vertices from getting too close to the surface.

Collision response. When a collision is detected, the vertex is projected back to the bone surface along the gradient direction:

$$\mathbf{p}_i^{\text{corrected}} = \mathbf{p}_i + (\epsilon - d(\mathbf{p}_i)) \cdot \nabla d(\mathbf{p}_i).$$

This projection can be formulated as a unilateral constraint in the XPBD framework, where the constraint is only active when penetration occurs.

Tunneling prevention. Fast-moving vertices may pass completely through thin bone structures between time steps, a phenomenon known as tunneling. Following Havlíček [Hav24], tunneling is detected by monitoring the gradient direction between consecutive frames. If the gradient direction changes by more than 135° , the vertex has likely tunneled through the bone and is repositioned accordingly.

6.3.2 Intra-bundle collision

Rods within the same muscle bundle must maintain their relative spacing during deformation. If rods collapse onto each other, the bundle loses its volume, and if they spread apart, the muscle cross-section expands unrealistically. Instead of employing explicit collision detection (e.g., the pill-based broadphase/narrowphase system of VIPER [Ang+19]), this thesis proposes a simpler and more effective approach using *slice distance constraints* that couple vertices across rods within the same cross-sectional slice.

Slice connectivity. The muscle decomposition methods (Kukačka and Hoang) produce fibers that pass through a series of cross-sectional slices. For each slice, every rod vertex is connected to its nearest and furthest neighbors on the same slice via standard XPBD distance constraints. The rest lengths are set to the initial distances between the connected vertices at the start of the simulation.

Nearest-neighbor connections. Connections to the K -nearest neighbors resist bundle collapse by penalizing distances shorter than the initial configuration. These constraints implicitly prevent rod overlap without requiring explicit collision detection or response, effectively solving the intra-bundle collision problem in a localized area through constraint-based spacing rather than geometric intersection tests. This avoids $O(n^2)$ cost of pairwise collision detection and integrates seamlessly into the existing constraint solver.

Furthest-neighbor connections. The idea behind the K -furthest neighbor connections is the concept of *small-world networks* from graph theory, where adding a few long-range connections to a locally connected network drastically reduces the

average path length between nodes, speeding up information propagation – see Figure 6.1. Close connection corrections would have to travel through many intermediate connections before reaching the opposite side of the cross-section, requiring more solver iterations and leading to slower convergence. The furthest-neighbor connections create long-range shortcuts that allow deformation to propagate across the entire cross-section in fewer iterations, achieving a similar effect to multi-grid methods and skip list-based acceleration structures, but without requiring any complex data structure.

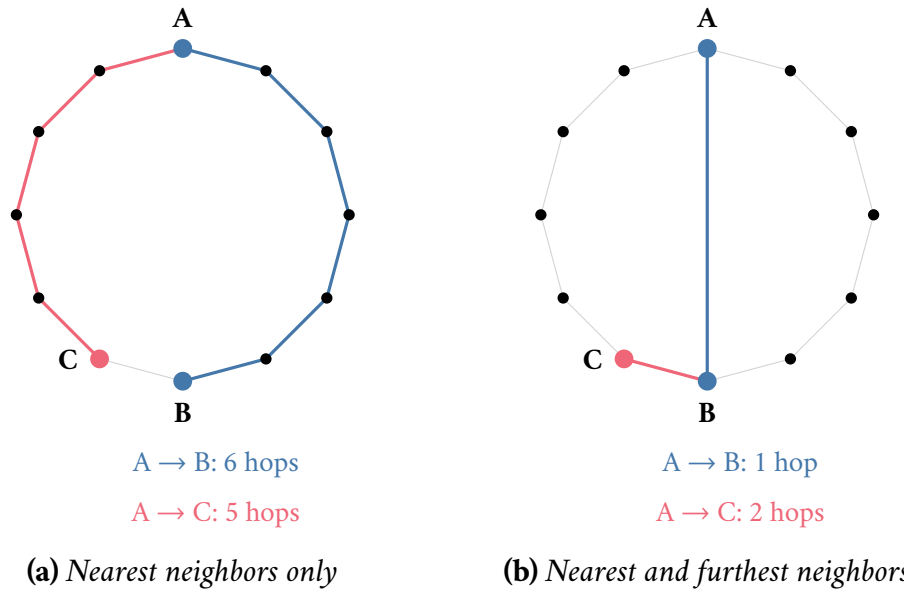


Figure 6.1: Simplified illustration of the small-world network principle applied to slice distance constraints. Without long range connections (a), information from node A requires 6 hops to reach node B and 5 to reach C. Adding a single furthest-neighbor connection (b) reduces $A \rightarrow B$ to 1 hop and $A \rightarrow C$ to 2 hops via the shortcut $A-B-C$

6.4 Activation-driven contraction

A key advantage of the rod-based approach is the natural integration of muscle activation into the simulation. This section describes how physiological activation signals from OpenSim are used to drive muscle contraction through modification of rod rest lengths.

6.4.1 Activation signal acquisition

Muscle activation levels are obtained from OpenSim’s Static Optimization Tool (SOT), following the pipeline established by Havlíček [Hav24]. The SOT estimates

muscle activations that produce the observed joint movements while minimizing a physiological cost function.

Static optimization. Given joint kinematics from motion capture or prescribed movement, the SOT solves an optimization problem to determine muscle activations $a_m \in [0, 1]$ for each muscle m :

$$\min_{a_m} \sum_{m=1}^n (a_m)^p$$

subject to the constraint that the muscle forces produce the required joint moments:

$$\tau_j = \sum_{m=1}^n a_m F_m^0 \cdot r_{m,j},$$

where τ_j is the net moment at joint j , F_m^0 is the maximum isometric force of muscle m , $r_{m,j}$ is its moment arm, and p is typically 2 (minimizing sum of squared activations).

Joint Control Unit. The Muscle Wrapping 2.0 framework organizes muscle activation through Joint Control Units (JCUs), which manage the muscles surrounding each joint [Hav24]. Each JCU stores:

- Sampled bone orientations covering the joint’s range of motion
- Corresponding muscle activations from SOT for each sample
- A kd-tree for efficient nearest-neighbor queries

Activation interpolation. During simulation, the current bone orientation is used to query the JCU for interpolated activation levels. A **radial basis function** (RBF) interpolation with Gaussian kernels provide smooth activation transitions:

$$\phi_n = e^{-F \cdot d_{q,n}^2},$$

where F is a fall-off rate parameter and $d_{q,n}$ is the distance in orientation space between the query and neighbor n . The interpolated activation for each muscle is the weighted average of its activations at the neighboring samples.

6.4.2 Rest length modification

The primary mechanism for activation-driven contraction is the modification of the rod segment rest lengths. When a muscle activates, its fibers shorten, which is simulated by reducing the reference lengths used by the stretch-shear constraint.

Contraction scaling. For a rod segment with original rest length $\hat{l}_e^0$ and current activation level $a \in [0, 1]$, the modified rest length is:

$$\hat{l}_e(a) = \hat{l}_e^0 \cdot (1 - \kappa \cdot a), \quad (6.6)$$

where $\kappa \in [0, 1]$ is the maximum contraction ratio. Based on physiological observations that sarcomeres contract to 50–70 % of their resting length [Hav24], typical values are $\kappa \in [0.3, 0.5]$, with $\kappa = 0.5$ allowing the muscle to shorten by up to 50 % at full activation.

Constraint modification. The stretch-shear constraint (Equation 6.3) uses the modified rest length $\hat{l}_e(a)$ instead of the original $\hat{l}_e^0$. This causes the constraint to pull vertices closer together, simulating active fiber shortening. The constraint solver naturally propagates this contraction through the rod, with the volume preservation constraint producing the characteristic bulging effect as the muscle contracts.

Per-segment vs. uniform activation. While the simplest approach applies uniform activation to all segments of a rod, physiological muscles may exhibit regional activation patterns. The rod-based representation naturally supports per-segment activation by storing individual rest lengths that can be modified independently. This capability is not currently exploited but provides flexibility for future extensions – see Chapter 9.

6.4.3 Contraction dynamics

The interplay between rest length modification and volume preservation produces the dynamic behavior of muscle contraction.

Volume-preserving bulging. When the stretch-shear constraint shortens a rod segment due to activation, the volume preservation constraint (Equation 6.5) automatically increases segment radius to maintain volume. Assuming approximately equal endpoint radii, the effective radius scales as:

$$r_i(a) = r_i^0 \sqrt{\frac{\hat{l}_e^0}{\hat{l}_e(a)}} = r_i^0 \sqrt{\frac{1}{1 - \kappa \cdot a}}.$$

At full activation ($a = 1$) with $\kappa = 0.5$, the radius increases by a factor of $\sqrt{2} \approx 1.41$, producing visible muscle bulging.

Propagation along the rod. Contraction does not occur instantaneously along the entire rod. The position-based solver propagates the rest length change through

iterative constraint projection, creating a wave-like contraction pattern that starts from the attachment points and moves toward the muscle belly. This behavior, while not explicitly modeled, emerges naturally from the constraint-based simulation and produces visually plausible dynamics.

Interaction with bend-twist. Active contraction may induce changes in rod curvature, particularly for pennate muscles where fibers run at an angle to the muscle’s line of action. The bend-twist constraint resists excessive curvature changes, providing passive resistance that contributes to the muscle’s overall mechanical behavior.

6.5 System overview

This section outlines the design goals, integration strategy, and overall simulation pipeline of the proposed rod-based muscle deformation method within the Muscle Wrapping 2.0 framework.

6.5.1 Design goals and constraints

The proposed method is designed to meet the following goals:

1. **Internal fiber representation** – represent muscles as bundles of volume-invariant Cosserat rods aligned with the muscle-tendon units, replacing the surface-only deformation of prior methods
2. **Volume preservation** – enforce local volume preservation at the rod level through the VIPER radius-stretch coupling, producing physiologically plausible bulging during contraction
3. **Framework compatibility** – integrate into the existing Muscle Wrapping 2.0 codebase without replacing the infrastructure developed by Červenka [Čer19] and Havlíček [Hav24], reusing their bone collision handling, activation pipeline, and VTK visualization
4. **Computational efficiency** – achieve near real-time performance (targeting 24+ FPS) by exploiting the parallel nature of per-rod constraint projection

6.5.2 Integration with Muscle Wrapping 2.0

The Cosserat rod simulation will extend the existing XPBD implementation rather than replacing it. Rod state will be added as member variables to the existing solver, and rod constraint projection is inserted into the main simulation loop alongside

the existing joint interaction system that provides activation signals. The three-layer architecture established by Červenka (OpenSim algorithm class, VTK filter class, and internals class) will be preserved, with new classes mirroring the existing pattern. Červenka's graph class will provide the bone SDF collision detection for rod vertices. The implementation details are described in Chapter 7.

6.5.3 Simulation pipeline

Each simulation frame proceeds through the following stages:

1. **Bone transform update** – bone positions and orientations are received from OpenSim, and fixed rod endpoints are updated to follow their respective bones
2. **Activation lookup** – the current bone orientation is used to query the Joint Control for interpolated muscle activation levels
3. **Rest length modification** – activation values are applied to modify the rest lengths of the stretch-shear constraints, simulating fiber contraction
4. **Position and orientation prediction** – rod vertex positions and edge quaternions are predicted using semi-implicit Euler integration
5. **Constraint projection** – the XPBD solver iteratively projects the stretch-shear, bend-twist, volume, and slice distance constraints
6. **Rod-bone collision** – rod vertices and edge samples are tested against bone SDFs and corrected to prevent penetration
7. **Velocity update** – linear and angular velocities are updated based on the position and quaternion changes

This pipeline is detailed in Chapter 7 (Algorithm 3).

Summary

This chapter presented a method for simulating muscle deformation using volume-invariant Cosserat rods within the XPBD framework. The key components of the proposed approach are:

- **Rod-based representation** – Muscle-tendon units are represented as bundles of VIPER-style rods, with each rod capturing a fiber path from origin to insertion. The discrete rod model stores positions and radii at vertices, with orientations (quaternions) defined per edge.

- **XPBD constraints** – Rod mechanics are governed by stretch and bend-twist constraints adapted from Kugelstadt and Schömer [KS16], while volume preservation is enforced through the radius-stretch coupling from VIPER [Ang+19]. Attachment constraints fix rod endpoints to bones.
- **Collision handling** – Rod-bone collisions use signed distance fields inherited from the existing framework, while intra-bundle spacing is maintained through slice distance constraints that connect rod vertices across cross-sectional slices.
- **Activation-driven contraction** – Muscle activation from OpenSim modifies rod rest lengths, causing fibers to shorten. The volume preservation constraint automatically produces the characteristic muscle bulging during contraction.

The method addresses the limitations of surface-based approaches identified in Chapter 5 by providing internal fiber representation, local volume preservation, and physiologically-driven activation. The following chapter describes the implementation of this method within the Muscle Wrapping 2.0 framework.

Implementation details

7

This chapter describes the implementation of the method proposed in Chapter 6 within the Muscle Wrapping 2.0 framework. While the previous chapter presented the theoretical formulation of each constraint, collision handler, and activation mechanism, this chapter focuses on how these components were realized in code, how they integrate with the existing codebase, and what engineering decisions were made along the way. The description loosely follows the chronological order of development, highlighting unexpected hurdles encountered during integration and the solution adopted.

7.1 Software architecture

The implementation builds upon the existing Muscle Wrapping 2.0 codebase, a C++17/20 project built with CMake and depending on VTK for visualization, Eigen for linear algebra, Discregrid for signed distance field computation, and OpenSim for musculoskeletal modeling. The primary application, `OsimMuscleGeneratorTool`, is an OpenSim plugin that orchestrates muscle decomposition, fiber generation, and deformation through a pipeline of VTK filters. The overall project architecture is described in Section 5.1 and illustrated in Figure 5.1.

7.1.1 Codebase organization

Each deformation algorithm in Muscle Wrapping 2.0 follows a three-layer pattern inherited from Červenka [Čer19]:

1. ***OpenSim algorithm class*** – registers the method as a selectable option in the `.xml` configuration file and creates the corresponding VTK filter. For this thesis, this is the `Hoang2026CosseratXPBDAlgorithm`.
2. ***VTK filter class*** – serves as the entry point for the deformation pipeline, receiving musculoskeletal geometry from VTK and delegating the actual sim-

ulation to lower-level classes. The corresponding class within Muscle Wrapping 2.0 is `vtkMusculoSkeletalSystemDeformationCosseratXPBDHoang`.

3. ***Internals class*** – declared as a friend class of the VTK filter, holding the data and the XPBD instances as well as the merged mesh storage. Here, this is `vtkMusculoSkeletalSystemDeformationCosseratXPBDHoangInternals`.

Below the VTK layer, the simulation logic resides in Havlíček’s XPBD class, which was extended to host rod state and rod constraint projection alongside the original surface-based constraints. Rod-specific data structures and constraint classes are implemented as standalone files in the `CosseratXPBD/` directory.

7.1.2 Source files

Table 7.1 lists the source files added or substantially modified by this thesis.

Table 7.1: *Source files added or modified by this thesis*

File	Purpose
<i>CosseratXPBD/ (new)</i>	
<code>CosseratRod.h/.cxx</code>	Discrete rod data structure
<code>RodStretchShearConstraint.h/.cxx</code>	Stretch and shear constraints
<code>RodBendTwistConstraint.h/.cxx</code>	Bend and twist constraint
<code>RodVolumeConstraint.h/.cxx</code>	Volume preservation constraint
<code>RodSliceDistanceConstraint.h/.cxx</code>	Inter-rod slice coupling
<i>XPBD/ (extended)</i>	
<code>XPBD.h/.cxx</code>	Main solver class
<code>pbd_debug.h/.cxx</code>	Debug visualization class
<i>VTK layer/ (new)</i>	
<code>Hoang2026CosseratXPBDAlgorithm.h/.cpp</code>	OpenSim algorithm class
<code>vtkMSSD...XPBDHoang.h/.cxx</code> ¹	VTK filter class
<code>vtkMSSD...XPBDHoangInternals.h/.cxx</code>	Friend internals class
<i>CosseratXpbdDeformation/ (new)</i>	
<code>ArcDeformation.cxx</code>	Validation experiment

7.2 Integration with Muscle Wrapping 2.0

Integrating Cosserat rod simulation into the existing framework required choosing how the new solver should coexist with the existing surface-based XPBD imple-

¹ Full name `vtkMusculoSkeletalSystemDeformationCosseratXPBDHoang`, abbreviated for space.

mentation. The key constraint was reuse: Červenka's graph class provides bone SDF collision detection and Havlíček's XPBD class provides the solver loop, and duplicating either would be impractical.

7.2.1 Choosing the integration approach

Three approaches were considered. The first was a standalone CosseratXPBD simulator with its own solver loop, independent of Havlíček's XPBD. This was prototyped early but quickly abandoned because it required duplicating the entire solver infrastructure (i.e., gravity integration, velocity updates, collision handling, and joint activation), all of which already existed in tested form in Havlíček's code.

The second approach was to keep a single XPBD class but split its implementation across two source files: `XPBD.cxx` for surface-based methods and a new `XPBD_rods.cxx` for rod-based methods. This improves file organization without changing the overall architecture and interfaces.

The third (and currently active) approach is to extend the existing XPBD class directly in the existing source file, adding rod state as member variables and rod constraint projection as additional steps within the existing solver loop. This was chosen for development speed and ease of integration. The file split approach is planned as a post-thesis cleanup step to improve maintainability.

7.2.2 The VTK filter and internals pattern

As described in Section 7.1.1, each deformation algorithm in Muscle Wrapping 2.0 is wrapped in a VTK filter class with a companion internals class. The VTK filter receives bone transforms from OpenSim, delegates deformation to the XPBD solver, and writes the deformed positions back into VTK polydata for downstream analysis.

The internals class holds the XPBD instance, the intermediate data object, and the merged polydata of all muscles and bones. This separation exists because the VTK filter must conform to the VTK pipeline interface (which restricts member visibility), while the actual simulation state lives in the friend internals class where it can be accessed freely.

A recurring source of complexity in the codebase is the data flow between VTK and the solver. Červenka's data class acts as an intermediary between VTK's polydata representation (points stored as flat `double*` arrays) and the Eigen-based vectors used by the XPBD solver. This means positions are copied from VTK into data, then mapped into Eigen vectors in XPBD, and finally copied back into VTK at the end of each frame. This roundabout pipeline is inherited technical debt from the original PBD implementation and could not be redesigned without modifying Červenka's and Kohout's code.

7.2.3 Data layout in the solver

When rods are registered in the XPBD solver (via `XPBD::set_rods()`), their per-rod data is unpacked into flat arrays owned by the solver, following the same pattern Havlíček established for surface-based constraints. Rod vertex positions are appended to the shared position array used by surface vertices, allowing them to participate in the existing collision handling without modifications. Other rod state variables (radii, quaternions, velocities, inverse masses, and inverse inertias) are stored in separate flat arrays, laid out contiguously across all rods.

7.3 Rod data structures

This section describes the `CosseratRod` class and how rods are constructed from the output of the muscle decomposition pipeline.

7.3.1 From fibers to rods

The starting point for rod construction is the Hoang decomposition method from the author's bachelor thesis [Hoa23]. The method generates internal fibers as polylines running from origin to insertion. In their original form, these fibers carry only positional information (i.e., infinitely thin lines with no cross-sectional geometry).

For this thesis, the decomposition class `vtkMAFMuscleDecompositionHoang` was updated to additionally output per-vertex radius values. During the Voronoi sampling phase, each fiber point on a muscle cross-section is assigned a radius based on its proximity to neighboring fibers. The radius is set to half the distance to the nearest neighbor, clamped by the distance to the muscle contour wall, and scaled by a packing coefficient and overflow factor. The tuning controls how much the muscle volume is filled out by the rods and how much they can overflow the muscle boundary. Figure 7.1 demonstrates the effect of the overflow factor during radius assignment.

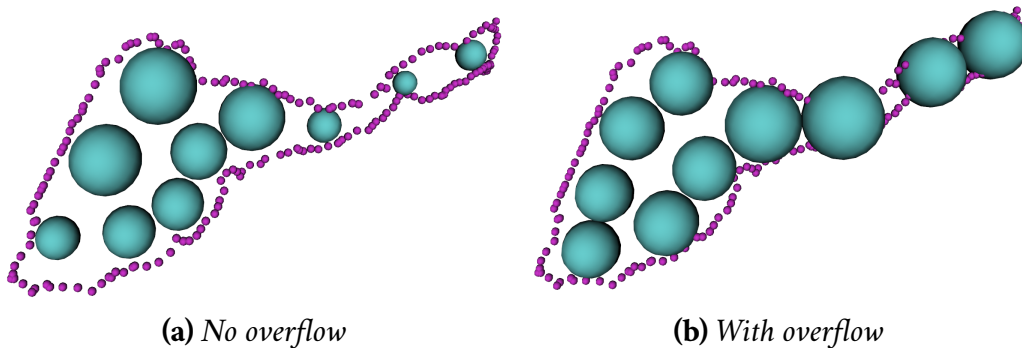


Figure 7.1: *Effect of the overflow factor on radius assignment*

This modification is important because it grounds the rod rest configuration in the actual geometry of the muscle rather than relying on uniform default values. The resulting radii vary along the rod and between rods, reflecting the natural variation of fiber spacing within the muscle.

7.3.2 The CosseratRod class

The CosseratRod class represents a single discrete volume-invariant Cosserat rod as described in Section 6.1. It follows the staggered grid approach of Kugelstadt and Schömer [KS16], with vertex-based radii from VIPER [Ang+19]. Listing 7.1 shows an abridged version of the class header.

Source code 7.1: *Abridged CosseratRod class header*

```

1 class CosseratRod {
2   CosseratRod(const std::vector<Eigen::Vector3d>& positions,
3               const std::vector<double>& radii,
4               double massPerVertex = 1.0);

6   // vertex data (N + 1 elements each)
7   std::vector<Eigen::Vector3d> mPositions;
8   std::vector<double> mRadii;
9   std::vector<Eigen::Vector3d> mVelocities;
10  std::vector<double> mInverseMasses;

12  // edge data (N elements each)
13  std::vector<Eigen::Quaterniond> mQuaternions;
14  std::vector<Eigen::Vector3d> mAngularVelocities;
15  std::vector<double> mInverseInertias;
16 };

```

A rod with N segments consists of $N + 1$ vertices and N edges. Positions and radii reside at vertices, while quaternion orientations reside at edges.

7.3.3 Initialization

The constructor accepts position and radii vectors from the decomposition output and initializes the remaining state variables – see Figure 7.2 for the initialization process. Edge orientations are computed using a **Bishop frame** (parallel transport) algorithm, i.e., the first edge quaternion aligns the reference direction $\mathbf{e}_3 = (0, 0, 1)$ with the first edge tangent, and each subsequent quaternion is obtained by applying minimal rotation that maps the previous tangent to the current one. This produces twist-minimizing frames even for curved rods.

The rest configuration (i.e., edge lengths, Darboux vectors, and frustum volumes) is then computed from the initial state and cached. These values serve as

targets for constraint projection during simulation: the stretch-shear constraint penalizes deviations from rest edge lengths, the bend-twist constraint penalizes deviations from rest Darboux vectors, and the volume constraint enforces rest frustum volumes. Finally, scalar inverse inertias are computed following Kugelstadt’s approximation [KS16] as $w_q \approx 1/\hat{l}_e^2$, giving shorter edges higher rotational responsiveness. If both endpoints of an edge are fixed, the inverse inertia is set to zero to prevent rotation.

Endpoints representing attachment points are fixed by setting their inverse mass to zero, which causes constraint corrections to skip them. Their positions are instead updated explicitly each frame by applying the current bone transform.

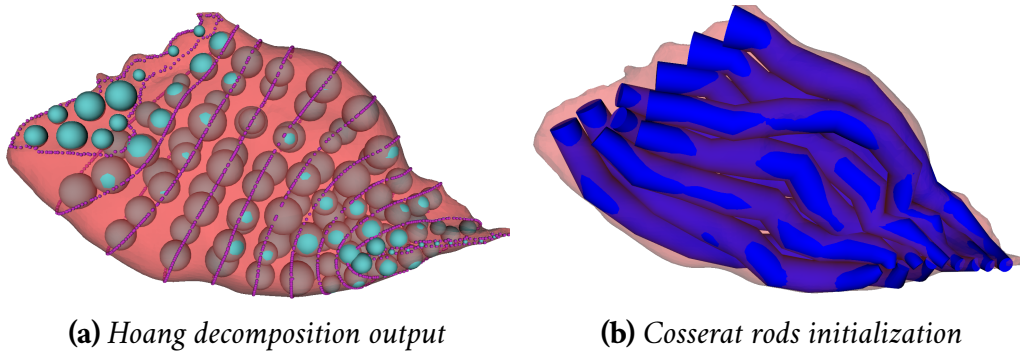


Figure 7.2: *Cosserat rod initialization process*

7.4 Constraint implementation

This section describes how the rod constraints formulated in Chapter 6 were implemented as C++ classes and why they could not use Havlíček’s existing constraint infrastructure.

7.4.1 Why standalone constraint classes

Havlíček’s surface constraints are built on a CRTP (Curiously Recurring Template Pattern) base class `constraint<Derived, N>`, which hardcodes N particles as `Vector3d` references and uses a scalar Lagrange multiplier λ . This design is efficient for surface-based constraints where every degree of freedom is a 3D position, but it is fundamentally incompatible with rod-based constraints, which mix different types of degrees of freedom: `Vector3d` positions, `Quaterniond` orientations, and scalar radii. For example, a stretch-shear constraint couples two positions and one quaternion and requires a *vector* Lagrange multiplier $\lambda \in \mathbb{R}^3$.

Rather than attempting to generalize the template (which would require modifying all existing surface constraints), each rod constraint was implemented as a standalone class. This follows the precedent set by Havlíček’s own `volumeConstraint`,

which also does not use the CRTP template because it operates on a variable number of particles. All four rod constraint classes share the same structure: a constructor that captures references to the relevant degrees of freedom, a `project()` method that performs one XPBD correction step, and a `reset_lambda()` method called at the start of each solver iteration.

7.4.2 Stretch-shear constraint

The `RodStretchShearConstraint` couples two vertex positions $\mathbf{p}_0$, $\mathbf{p}_1$ and one edge quaternion $\mathbf{q}$, following Kugelstadt equations 31 and 34–37 [KS16] as formulated in Equation 6.3. It uses a vector Lagrange multiplier $\lambda \in \mathbb{R}^3$, and the XPBD correction distributes updates to both positions and the quaternion, weighted by their inverse masses and inverse inertia. The rest length $\hat{l}_e$ can be modified at runtime to simulate activation-driven contraction (see Section 7.6).

7.4.3 Bend-twist constraint

The `RodBendTwistConstraint` couples two adjacent edge quaternions $\mathbf{q}_0$, $\mathbf{q}_1$, penalizing deviations of the current Darboux vector from its rest value following Kugelstadt and Schömer [KS16] (equations 32, 33, and 38–40). The constraint value and sign factor s are defined in Equation 6.4. The correction rotates both quaternions toward the rest relative orientation weighted by their inverse inertias.

7.4.4 Volume constraint

The `RodVolumeConstraint` preserves the truncated cone volume of a single rod segment as formulated in Equation 6.5. The concept of treating the radius as an additional degree of freedom originates from VIPER [Ang+19], but their formulation uses an energy-based approach with Gauss-Newton optimization. The constraint presented here reformulates volume preservation as an XPBD constraint using the frustum formula, coupling two positions $\mathbf{p}_0$, $\mathbf{p}_1$ and two radii r_0 , r_1 with a scalar Lagrange multiplier λ .

In practice, position inverse masses for this constraint are set to zero, forcing all volume corrections through the radii. This prevents the volume constraint from fighting the stretch-shear constraint over edge lengths and is physiologically motivated. Real muscles change their cross-sectional shape to preserve volume, not their fiber length. Radii are clamped to a minimum value after each projection to prevent numerical instability when a segment is stretched far from its rest length.

Figure 7.3 demonstrates the effect on the iliopsoas muscle during hip flexion, the right image shows visible bulging of individual rods.

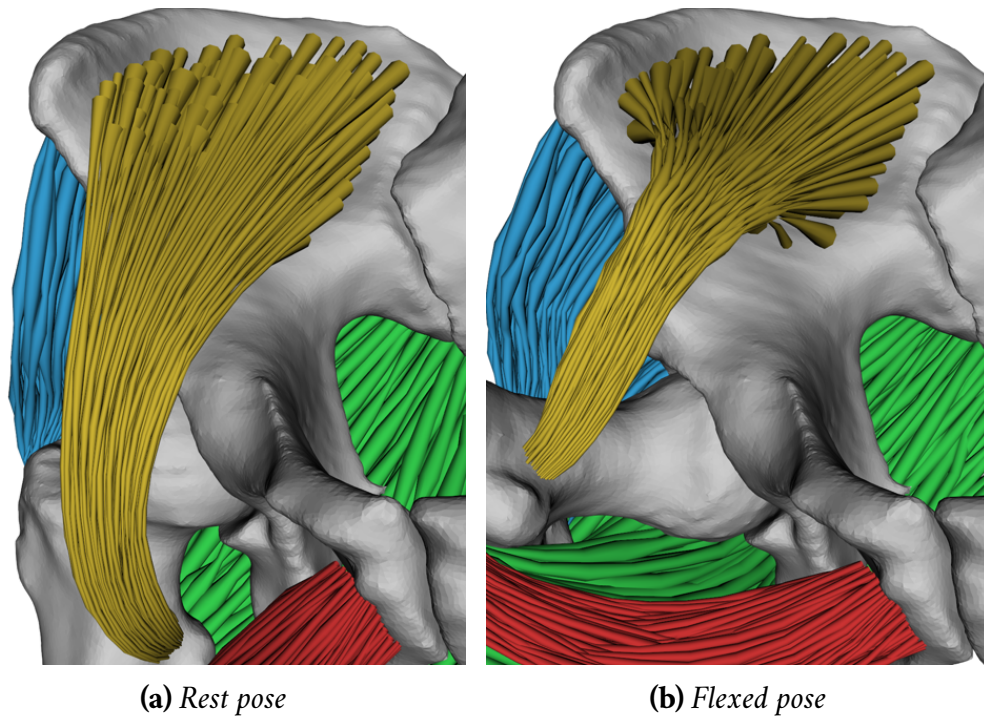


Figure 7.3: Volume constraint effect on iliopus (yellow) during hip flexion

7.4.5 Slice distance constraint

The RodSliceDistanceConstraint is a standard XPBD distance constraint between two vertices on different rods at the same cross-sectional slice. For each vertex, connections are made to the $K = 6$ nearest and $K = 2$ furthest neighbors on the same slice. Figure 7.4 shows the effect of the slice distance constraint on the gluteus maximus during hip flexion. Without the constraint, rods collapse onto each other and separate into multiple disconnected clusters.

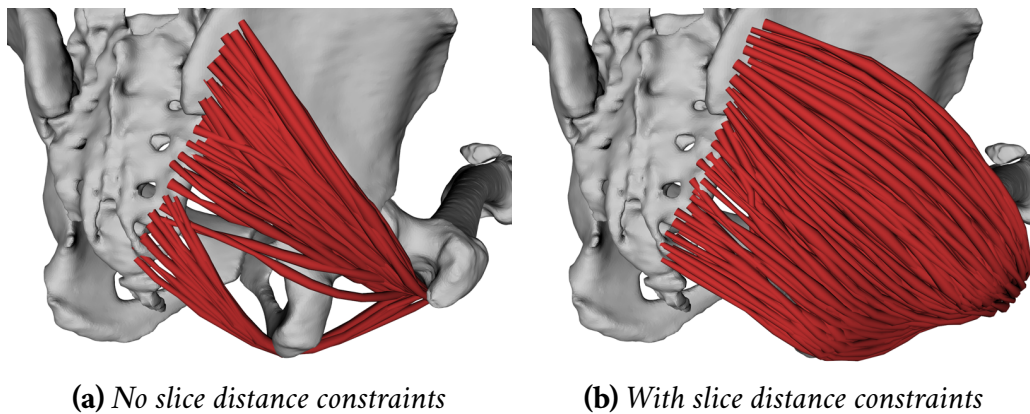


Figure 7.4: Slice distance constraint effect on gluteus maximus during hip flexion

7.4.6 Compliance time step convention

All four constraint classes define a constant `COMPLIANCE_TIME_STEP = 1.0` that is used when computing the scaled compliance $\tilde{\alpha} = \alpha / \Delta t^2$. This corresponds to `s_time_diff` in Havlíček’s CRTP constraint base class, where it is documented as an intentional simplification. Since Δt influences the compliance denominator quadratically, using a realistic time step (e.g., 10^{-3}) forces compliance values into extreme ranges to achieve reasonable behavior. Using a fixed time step $\Delta t = 1.0$ allows compliance values to be more intuitive and easier to tune. The actual simulation time step `m_time_diff = 0.001` is used for velocity integration and position prediction. The constant is defined per constraint class rather than globally, allowing independent tuning if needed.

7.5 Collision handling

Collision handling in the rod-based simulation can be divided into three categories: collision between rods within the same muscle bundle, collision between rods and bones, and collisions between rods of different muscles.

7.5.1 Intra-bundle collision

Collisions within the same bundle are implicitly handled by the slice distance constraint described in Section 7.4.5. The $K = 6$ nearest neighbor connections preserve the cross-sectional spacing between rods on each slice, preventing rod overlap without the need for explicit collision detection or response. This approach avoids the cost of a broadphase search within each bundle and produces stable results as long as the initial rod configuration is reasonably well-spaced. In practice, the Hoang decomposition method produces well-distributed fibers, so the slice distance constraint is sufficient to prevent intra-bundle collisions without additional measures.

7.5.2 Rod-bone collision

Rod-bone collision handling reuses Červenka’s *signed distance field* (SDF) infrastructure without modification and extends it with edge-level sampling to address a failure case specific to rod geometry. Two collision handlers were implemented. Vertex-level collision handling treats each rod vertex independently, while edge-level collision handling additionally samples points along the edge to catch cases where the edge penetrates a bone even though both endpoints lie outside it.

Vertex-level collision. For each non-fixed rod vertex, the solver queries Červenka’s `collision_detection::find()` method, passing the previous and current posi-

tions. If a collision is detected, the offending vertex is pushed out along the SDF gradient. This is identical to how surface mesh vertices are handled in Havlíček’s implementation and requires no new code.

Edge-level collision. Rod edges can be significantly longer than surface mesh edges, making it possible for both endpoints to lie outside a bone while the edge itself penetrates it. This is especially problematic for long muscles like the *rectus femoris* and near anatomical features with thin structures or holes, such as the *obturator foramen*. To address this, additional points are sampled along each rod edge segment at intervals controlled by `ROD_EDGE_COLLISION_SAMPLE_SPACING_M` (default 6.5 mm). For each sample, the SDF is queried. If the sample is inside the bone, a correction is computed and distributed to the edge’s two endpoints using barycentric weights $1 - t$ and t , where t is the sample’s interpolation parameter along the edge.

Figure 7.5 shows the absence of edge-level collision handling. The gluteus maximus fibers penetrate the bone geometry although the rod vertices remain outside the pelvis SDF.

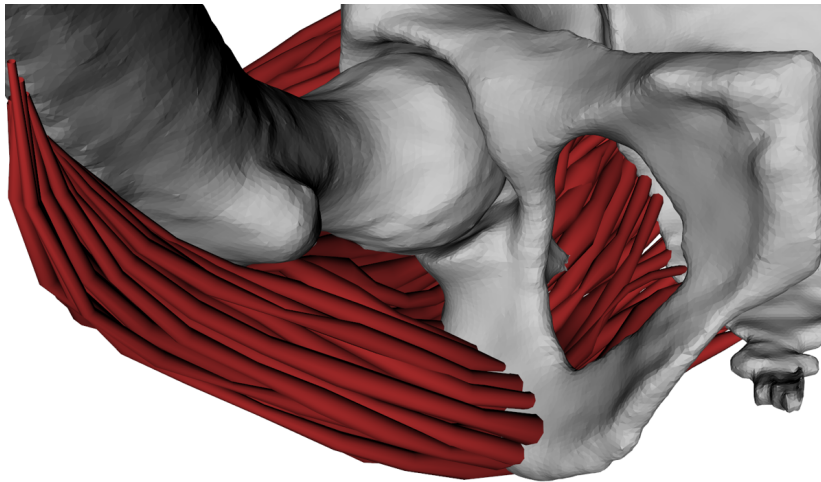


Figure 7.5: *Gluteus maximus penetrating the pelvis without edge-level collision handling*

Scalar interpolation: a failed approach. Before arriving at the sampling approach, an alternative was explored – caching the SDF distance at each rod vertex and linearly interpolating along edges to detect edge penetration. This would have been cheaper, as it avoids repeated SDF queries for sample points. However, linear interpolation of SDF values fails for anatomical structures with topological holes. Near the *obturator foramen*, both endpoints can have positive SDF values (outside the bone), so the interpolated values remain positive along the entire edge, even though the edge passes through the hole. This approach was implemented, tested, and discarded in favor of the edge-level sampling method.

7.5.3 Inter-muscular collision

Collisions between rods belonging to different muscle bundles are not handled in the current implementation. This would require a broadphase spatial data structure to efficiently identify candidate rod pairs across muscle boundaries, followed by a narrowphase capsule-based closest-point computation. The infrastructure for this was designed but not completed within the scope of this thesis and remains as future work (see Chapter 9).

7.6 Activation pipeline

Muscle activation drives rod contraction by modifying the rest lengths of stretch-shear constraints. The pipeline consists of three stages: acquiring the raw activation signal from OpenSim, smoothing it to avoid numerical artifacts, and applying the resulting contraction to each rod.

7.6.1 Activation signal acquisition

Raw activation values $a \in [0, 1]$ for each muscle are obtained from Havlíček’s joint interaction system, which interpolates activations from OpenSim’s *Static Optimization Tool* (SOT) output as described in Section 6.4.1. Each rod is associated with a single muscle name at construction time, and the solver looks up the corresponding activation from the joint that manages that muscle. All rods belonging to the same muscle receive the same activation value.

7.6.2 Activation smoothing

Applying raw activation values directly to the rod rest lengths caused visible twitching during simulation. The root cause is the temporal sparsity of the activation samples. OpenSim’s SOT produces activation values at relatively coarse time intervals, and the abrupt jumps between values cause sudden rest-length changes that propagate as jerky motion.

Two smoothing mechanisms are applied in sequence. First, the raw activation is filtered through an exponential moving average (EMA):

$$a_{\text{smooth}}^{(n)} = a_{\text{smooth}}^{(n-1)} + \alpha_{\text{EMA}} \cdot (a_{\text{raw}} - a_{\text{smooth}}^{(n-1)}),$$

where $\alpha_{\text{EMA}} = 0.1$ (ACTIVATION_SMOOTHING_ALPHA) controls the smoothing strength. Second, the per-frame change in activation is clamped to a maximum value of 0.05 (MAX_ACTIVATION_DELTA), preventing any single frame from producing a drastic contraction change. These values were determined experimentally to eliminate twitching while preserving the overall contraction dynamics.

Figure 7.6 shows the effect of EMA smoothing and rate clamping on raw activation signals from OpenSim for *iliacus* during a flexion movement.

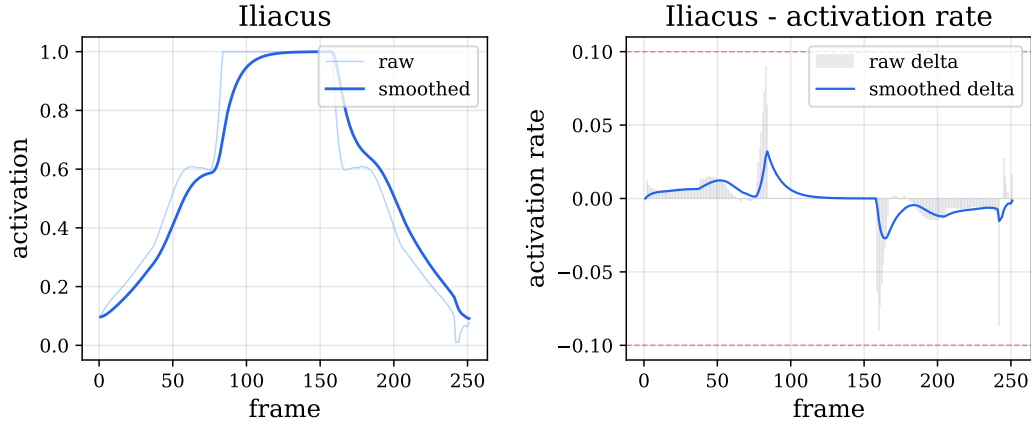


Figure 7.6: Effect of EMA smoothing and rate clamping on raw activation signals from OpenSim over a 250 frame motion sequence. Left: raw (light blue) versus smoothed (dark blue) activation curves. Right: corresponding frame-to-frame deltas showing that the smoothed signal stays well within the maximum delta threshold, while the raw signal exhibits sharp spikes that manifest as visible twitching in the simulation

7.6.3 Rest length modification

The smoothed activation drives contraction by scaling the rest length of each stretch-shear constraint:

$$\hat{l}_e(a) = \hat{l}_e^0 \cdot (1 - \kappa \cdot a),$$

where $\hat{l}_e^0$ is the original rest length at initialization and $\kappa = 0.4$ is the contraction ratio, meaning a fully activated muscle contracts to 60 % of its rest length. This value was based on physiological estimates of maximum muscle shortening (see Section 6.4.3).

The volume constraint compensates the length contraction by increasing the radii of the affected segments, producing the characteristic bulging. This coupling between activation (shortening rest lengths), stretch-shear (enforcing the new rest lengths), and volume preservation (expanding radii to compensate) is the core mechanism of the proposed method.

7.7 Solver loop

This section describes how the rod constraints integrate into the existing XPBD solver loop. The `execute()` method, called once per frame, follows the standard XPBD pattern from Macklin et al. [MMC16]: predict positions, iteratively project constraints, then update velocities.

7.7.1 Extended solver loop

The rod-based simulation replaces the surface-based constraint projections from Havlíček’s solver with rod-specific ones. When rod constraints are enabled (i.e., when `ENABLE_ROD_CONSTRAINTS = true`), the surface-based distance, dihedral angle, and volume constraints are disabled, as are surface-bone and surface-surface collision handling. Algorithm 3 shows the structure of a single solver iteration with rod constraints.

Algorithm 3: Rod-based XPBD solver loop (one frame)

```

1 predict  $\tilde{x} \leftarrow x_n + \Delta t v_n + \Delta t^2 M^{-1} f_{ext}$ 
2 predict  $\tilde{q} \leftarrow q_n + \frac{\Delta t}{2} \omega_n q_n$ 
3 update rod rest lengths based on current activation
4 initialize  $\lambda \leftarrow 0$  for all constraints
5 while  $i < solverIterations$  do
6   apply joint interactions
7   forall rods (in parallel) do
8     project stretch-shear constraints
9     project bend-twist constraints
10    project volume constraints
11  end
12  project slice distance constraints
13  handle rod-bone collisions
14   $i \leftarrow i + 1$ 
15 end
16 update velocities  $v_{n+1} \leftarrow (x_{n+1} - x_n) / \Delta t$ 
17 update angular velocities  $\omega_{n+1} \leftarrow 2(q_{n+1} \bar{q}_n) / \Delta t$ 

```

Havlíček’s joint interactions are retained because they provide the activation signals consumed by the rod contraction pipeline – see Section 7.6.

7.7.2 Constraint projection order

Within each solver iteration, per-rod constraints (stretch-shear, bend-twist, and volume) are projected together per rod. This means that for a given rod, the solver projects all stretch-shear constraints, then all bend-twist constraints, then all volume constraints, before moving to the next rod. This ordering was chosen because the three constraint types within a rod share vertices and quaternions, so projecting sequentially avoids write conflicts and allows the constraints to build on each other’s corrections within the same iteration.

Slice distance constraints are projected separately after all per-rod constraints, since they couple vertices across different rods and cannot be parallelized in the same per-rod loop.

7.7.3 Compile-time feature flags

Individual constraint types and processing modes can be toggled with the help of `static constexpr bool` flags declared in `XPBD.h`, as shown in Listing 7.2. The flags are organized hierarchically so that disabling the top-level switch at line 2, i.e., `ENABLE_ROD_CONSTRAINTS`, disables all rod features (lines 3–6). Code paths are guarded by `if constexpr` (lines 9, 13) statements, ensuring disabled features produce no runtime overhead.

Source code 7.2: *Compile-time feature flags and their usage*

```
1 // flag declarations in XPBD.h
2 static constexpr bool ENABLE_ROD_CONSTRAINTS = true;
3 static constexpr bool ENABLE_ROD_PARALLEL_PROCESSING =
    ENABLE_ROD_CONSTRAINTS && true;
4 static constexpr bool ENABLE_ROD_STRETCH_SHEAR =
    ENABLE_ROD_CONSTRAINTS && true;
5 static constexpr bool ENABLE_ROD_BEND_TWIST =
    ENABLE_ROD_CONSTRAINTS && true;
6 static constexpr bool ENABLE_ROD_VOLUME =
    ENABLE_ROD_CONSTRAINTS && true;

8 // example usage in XPBD.cxx
9 if constexpr (ENABLE_ROD_STRETCH_SHEAR) {
10     add_rod_stretch_shear_constraints();
11 }
12 if constexpr (ENABLE_ROD_BEND_TWIST) {
13     add_rod_bend_twist_constraints();
14 }
```

7.8 Parallelization

Rod constraint projection is parallelized at the per-rod level using a utility template function `mw2::for_each()` as shown in Listing 7.3. When the template parameter `Parallel = true`, elements are processed using `std::execution::par_unseq`. If false, a simple range-based for loop is used instead. Since the parameter is resolved at compile time (see example usage at lines 19–24), switching between parallel and sequential execution produces no runtime overhead. Within the XPBD solver, this is controlled by the `ENABLE_ROD_PARALLEL_PROCESSING` flag from Listing 7.2.

Source code 7.3: `mw2::for_each()` and usage

```

1 namespace mw2 {
2 template <bool Parallel>
3 void for_each(std::ranges::range auto&& range, auto&& func) {
4     if constexpr (Parallel) {
5         std::for_each(std::execution::par_unseq,
6                       std::ranges::begin(range),
7                       std::ranges::end(range),
8                       std::forward<decltype(func)>(func));
9     }
10    else {
11        for (auto&& element : range) {
12            func(element);
13        }
14    }
15 }
16 } // namespace mw2

18 // usage in XPBD.cxx
19 mw2::for_each<ENABLE_ROD_PARALLEL_PROCESSING>(
20     m_rod_stretch_shear_constraints,
21     [](auto& c) {
22         c.reset_lambda();
23     }
24 );

```

The key insight enabling parallelization is the embarrassingly parallel nature of per-rod constraint projection: constraints belonging to different rods share no degrees of freedom and can be solved independently without synchronization. This stands in contrast to Havlíček’s surface-based parallelization, which required greedy graph coloring to identify independent constraint groups within the same shared mesh topology – a significantly more complex approach that still could not fully eliminate sequential processing. Rod-bone collision handling is likewise parallelized per rod, since each rod’s collision response only modifies its own vertices. Slice distance constraints are the only exception, as they couple vertices across different rods and are projected sequentially after all per-rod constraints. However, constraints on different slices do not share vertices and could be parallelized in future work (see Chapter 9).

The parallelization yields a $4.7\times$ speedup for the per-rod constraint projection and a $6.5\times$ speedup for rod-bone collision handling on an 8-core/16-thread AMD Ryzen PRO 6850U laptop CPU. The detailed performance analysis is presented in Section 8.4.

7.9 Validation infrastructure

To validate the constraint formulations independently of the full musculoskeletal pipeline, a standalone validation experiment was developed alongside a statistical analysis pipeline.

7.9.1 Quarter-circle arc experiment

The ArcDeformation program listed in Table 7.1 is a self-contained validation experiment that creates a bundle of concentric quarter-circle arc rods, applies a prescribed angular deformation (closing from 90° to 45°) as seen in Figure 7.7, and exports the resulting geometry and measurements as `.vtk` and `.csv` files.

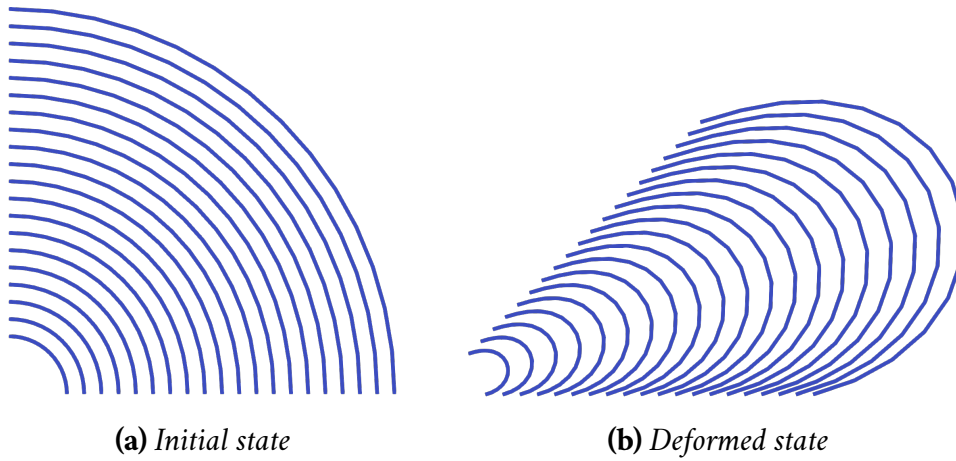


Figure 7.7: Arc deformation experiment, initial state (left) and deformed state (right). The origin $[0, 0]$ serves as the joint center, while the x and y axes represent the bones. Fixed rod endpoints are anchored along the axes and are driven by the bone transforms to produce the desired angular deformation

The experiment runs four configurations, each enabling a different combination of rod constraints and activation-driven contraction:

1. **Baseline** – stretch-shear and bend-twist constraints only
2. **Volume only** – add volume preservation
3. **Activation only** – add activation-driven contraction
4. **Full** – all constraints and activation enabled

By comparing these configurations, the contributions of each constraint type can be isolated. The experiment operates entirely on the `CosseratRod` and constraint

classes without depending on VTK filters, OpenSim, or the graph/data infrastructure, providing a controlled environment for validating the core mechanics of the rod simulation.

7.9.2 Statistical analysis pipeline

The exported .csv data is processed by a Python analysis pipeline, which operates in three modes: arc deformation analysis (computing volume ratios, segment length distributions, and shape preservation metrics), activation signal diagnosis (comparing raw vs smoothed activation curves), and fiber length comparison between the surface-based and rod-based approaches via OpenSim .sto files. The analysis produces figures and statistical tables used in the results chapter. The experimental setup, hypotheses, and statistical methodology are described in detail in Chapter 8.

Results and evaluation

8

This chapter presents the experimental evaluation of the proposed method. The evaluation proceeds in three stages: a controlled validation experiment on a synthetic geometry (Section 8.2), a musculoskeletal simulation of hip flexion comparing the rod-based approach with Havlíček’s surface-based method (Section 8.3), and a performance analysis measuring computational cost and parallelization gains (Section 8.4).

8.1 Experimental setup

All experiments were conducted on a ThinkPad P16s Gen 1 laptop (product number: 21CK0031CK) with an AMD Ryzen PRO 6850U processor (8 cores, 16 threads) and 16 GB of RAM, running Windows 11 Pro. The simulation code was compiled with Microsoft Visual Studio 2026 (MSVC v18.5.1) in Release mode.

The quarter-circle arc experiment (Section 8.2) runs as a standalone C++ program (`ArcDeformation.cxx`) as described in Section 7.9. The hip flexion experiment (Section 8.3) runs the full Muscle Wrapping 2.0 pipeline, using OpenSim to drive bone transforms and activation signals. Both experiments share the solver parameters listed in Table 8.1.

Table 8.1: *Solver parameters shared across experiments*

Parameter	Value	Description
Time step Δt	0.001 s	Simulation time step
Velocity damping	0.9	Per-frame velocity scaling
Stretch-shear compliance	0.0	Infinite stiffness
Bend-twist compliance	0.0	Infinite stiffness
Volume compliance	0.0	Infinite stiffness
COMPLIANCE_TIME_STEP	1.0	Compliance time step scaling
VOLUME_POSITION_WEIGHT	0.0	Volume corrects radii only

The zero compliance values enforce hard constraints, ensuring that stretch-shear, bend-twist, and volume preservation are strictly maintained. Setting the `VOLUME_POSITION_WEIGHT = 0.0` forces the volume constraint to distribute all corrections to the radii rather than positions, so as not to compete with the stretch-shear constraint.

8.2 Quarter-circle arc experiment

The quarter-circle arc experiment validates the rod constraint system in a controlled geometric setting. A bundle of concentric circular arc rods is created with fixed end-points anchored to the coordinate axes, representing two bones meeting at a joint. The deformation consists of closing the arc from 90° to 45° over 100 simulation steps, producing a contraction ratio of $\kappa = 1 - \frac{45^\circ}{90^\circ} = 0.5$ – a physiologically plausible maximum shortening for skeletal muscle – see Section 6.4.3.

The experiment uses 20 concentric rods with 15 vertices each (14 segments per rod), spanning arc radii from 0.15 to 1.0 units and each rod has a cross-sectional radius of 0.005 units. The solver runs 20 iterations per time step.

The four configurations described in Section 7.9.1 are evaluated against three research questions, each targeting a different aspect of the simulation.

8.2.1 RQ1: Vertex distribution uniformity

The first research question asks whether rod vertices remain uniformly distributed along the arc during deformation, validating that the Cosserat rod representation produces well-behaved simulations without vertex clustering or excessive stretching – both cases are visualized in Figure 8.1.

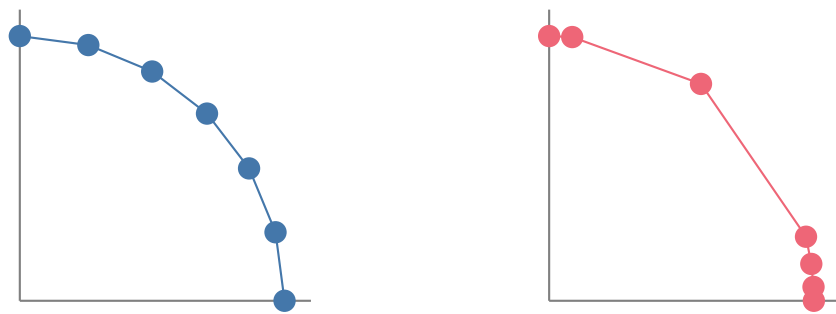


Figure 8.1: Vertex distribution along a deformed arc rod. Uniform spacing (left) produces well-behaved segments. Collapsed vertices (right) create short degenerate segments near the fixed endpoints and long overstretched segments elsewhere

Metric. For each rod with N vertices and $N - 1$ segments, the **normalized segment-length ratio** for segment i is defined as:

$$\hat{r}_i = \frac{d_i \cdot (N - 1)}{L}, \quad (8.1)$$

where d_i is the segment length and L is the total arc length. For perfectly uniform spacing, $\hat{r}_i = 1.0$ for all segments. The mean of $\hat{r}_i$ should be close to 1.0, so the **coefficient of variation** (CV) serves as the per-rod uniformity metric instead.

Results. Figure 8.2 shows the distribution of normalized segment-length ratios at the final step. All four configurations keep the ratios close to 1.0, confirming that rod vertices do not collapse or separate during deformation.

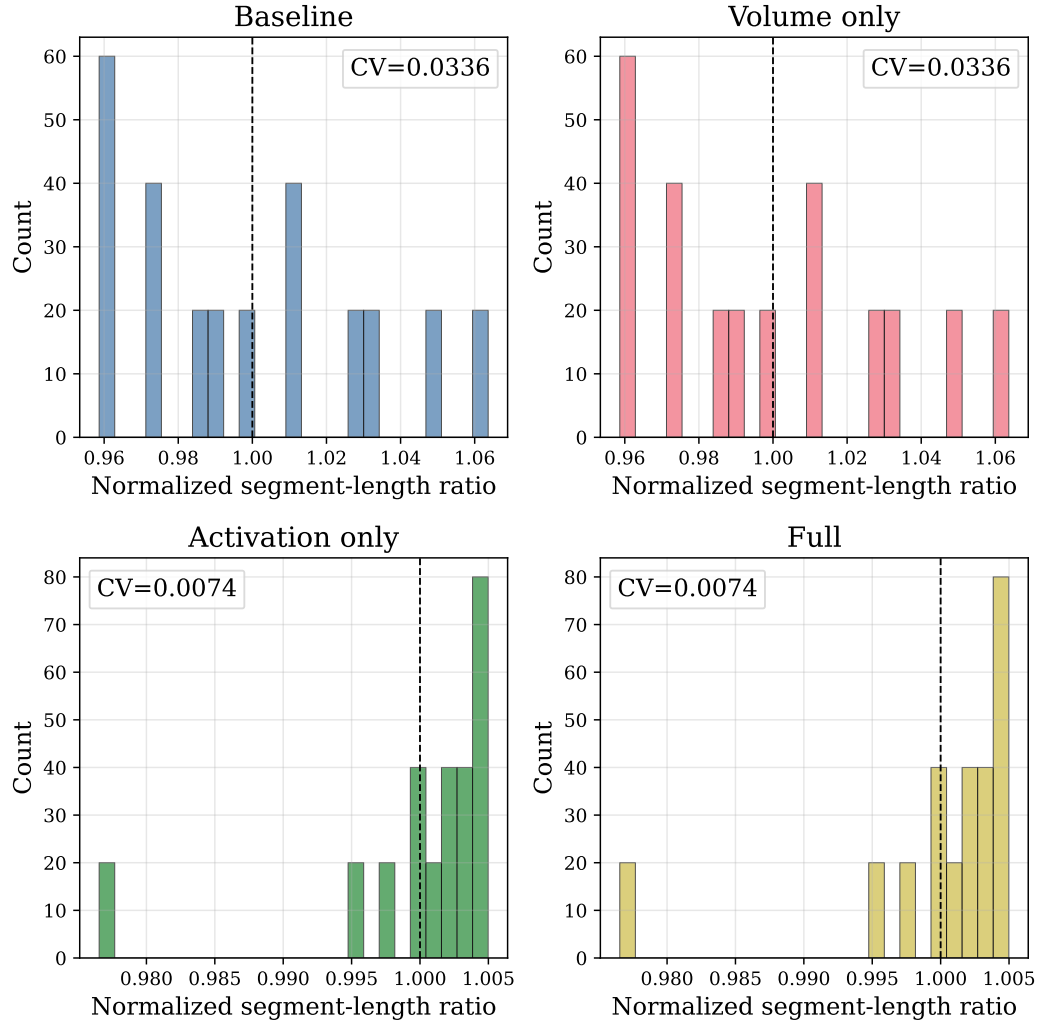


Figure 8.2: Distribution of normalized segment-length ratios at the final deformation step across all rods, per configuration. The dashed line marks the ideal value of 1.0, while CV is the mean coefficient of variation across all rods

However, a stark contrast emerges between passive and active configurations. The baseline and volume-only configurations exhibit a wider spread ($CV = 0.036$, ratios from 0.96 to 1.07), while the activation-only and full configurations are visibly tighter ($CV = 0.0074$, ratios between 0.975 and 1.005).

This improvement under activation is explained by the contraction mechanism. In the passive configurations, the solver must accommodate the angular deformation purely through constraint projection, and the resulting edge lengths drift from their rest values. With activation enabled, the rest lengths are shortened to match the target arc length via the contraction ratio κ , so the stretch-shear constraint enforces lengths that are already close to the deformed configuration. The solver thus converges quicker and produces more uniform spacing.

8.2.2 RQ2: Shape preservation under activation

The second research question asks whether activation-driven contraction is necessary for shape preservation. Without activation, the rods retain their original rest lengths during deformation. Each rod's shape is evaluated against an analytical reference arc at the rod's original radius and the current target angle of 45° .

Metric. Each vertex's displacement from the analytical reference is decomposed into polar components:

$$\begin{aligned}\varepsilon_r &= \|\mathbf{p}\| - R, \\ \varepsilon_\theta &= \text{atan2}(p_y, p_x) - \theta_{\text{ref}},\end{aligned}$$

where R is the rod's arc radius, $\mathbf{p}$ is the simulated vertex position, and θ_{ref} is the expected angular position on the reference arc. The radial error ε_r measures how far the vertex drifts from the correct radius, while the angular error ε_θ captures how much the vertex deviates along the arc. Both components are illustrated in Figure 8.3. The mean Euclidean displacement and the Fréchet distance (a curve similarity metric) are computed per rod.

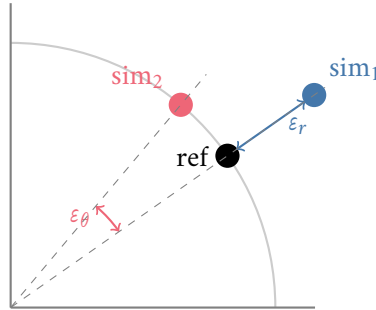


Figure 8.3: Polar error decomposition. Point sim_1 (blue) shows a radial error ε_r – correct angular position but wrong distance from the origin, while point sim_2 (red) shows an angular error ε_θ – correct radius but offset along the arc

Results. Figure 8.4 compares simulated and analytical rod arc position for a representative middle rod (the tenth rod out of 20):

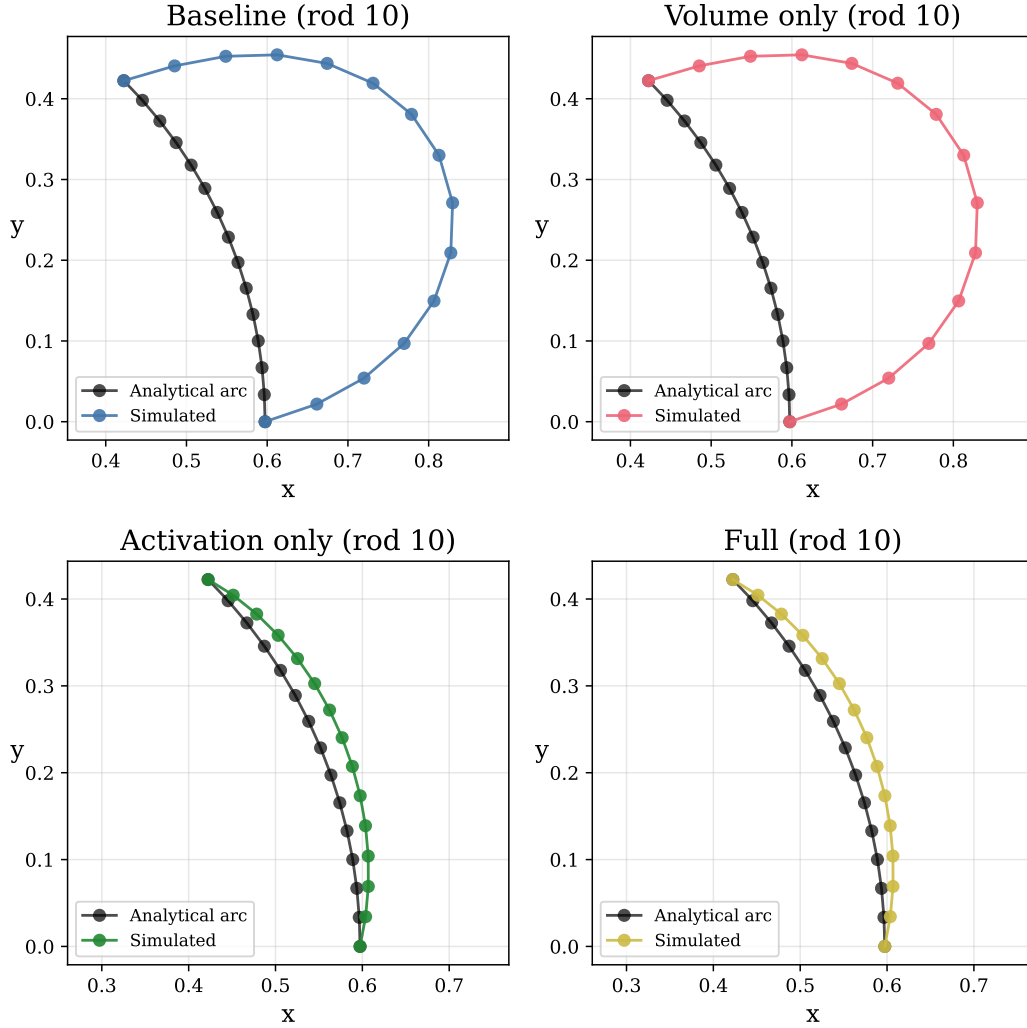


Figure 8.4: Simulated (colored) vs. analytical arc (black) positions for rod 10 at the final deformation step, per configuration. The passive configurations (baseline and volume-only) show large outward deviation because the rod retains its original rest length, which is too long for the shorter 45° arc. The active configurations (activation-only and full) shorten the rest length which keeps the rod close to the reference arc

Figure 8.5 illustrates the decomposition of the displacement into radial and angular components. The passive configurations (baseline and volume-only) exhibit radial errors up to 0.28 (vertices are pushed outward from the arc) and angular errors up to 2.5° . The active configurations (activation-only and full) reduce both deviations by an order of magnitude. Both the passive and active configurations overlap exactly, confirming that the volume constraint does not affect vertex positions when the `VOLUME_POSITION_WEIGHT = 0.0`.

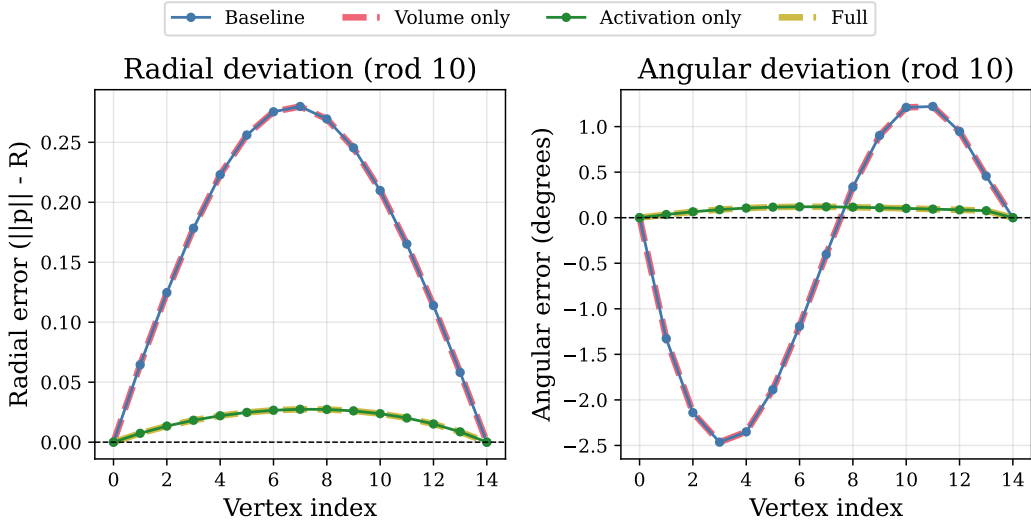


Figure 8.5: Per-vertex radial (left) and angular (right) errors at the final deformation step for rod 10. The passive and active configurations show a stark difference. In the radial deviation plot, the passive configurations (blue and pink) show visibly more bulging than the active configurations (green and yellow), the ideal is zero error (dashed line). In the angular deviation, the passive configurations show negative errors for the first half of the rod and positive errors from vertex 7 onward

The **Fréchet distance** measures the similarity between two curves as the maximum displacement between corresponding points. Here, vertex correspondence is known at construction time (vertex i of the simulated rod corresponds to vertex i of the analytical arc), so the Fréchet distance simplifies to the maximum per-vertex Euclidean distance – see Figure 8.6.

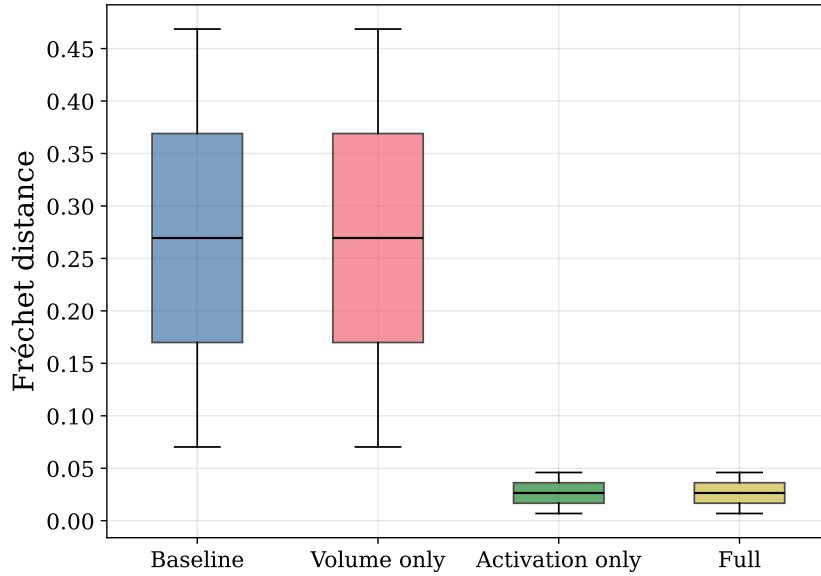


Figure 8.6: Fréchet distance across all 20 rods per configuration

The passive configurations show a median Fréchet distance of 0.27, while for the active configurations it remains below 0.05, confirming that the shape difference observed for rod 10 is consistent across the entire bundle.

The final deformed state of all four configurations is visualized in Figure 8.7 as rendered in ParaView, a popular open-source scientific visualization tool based on VTK.

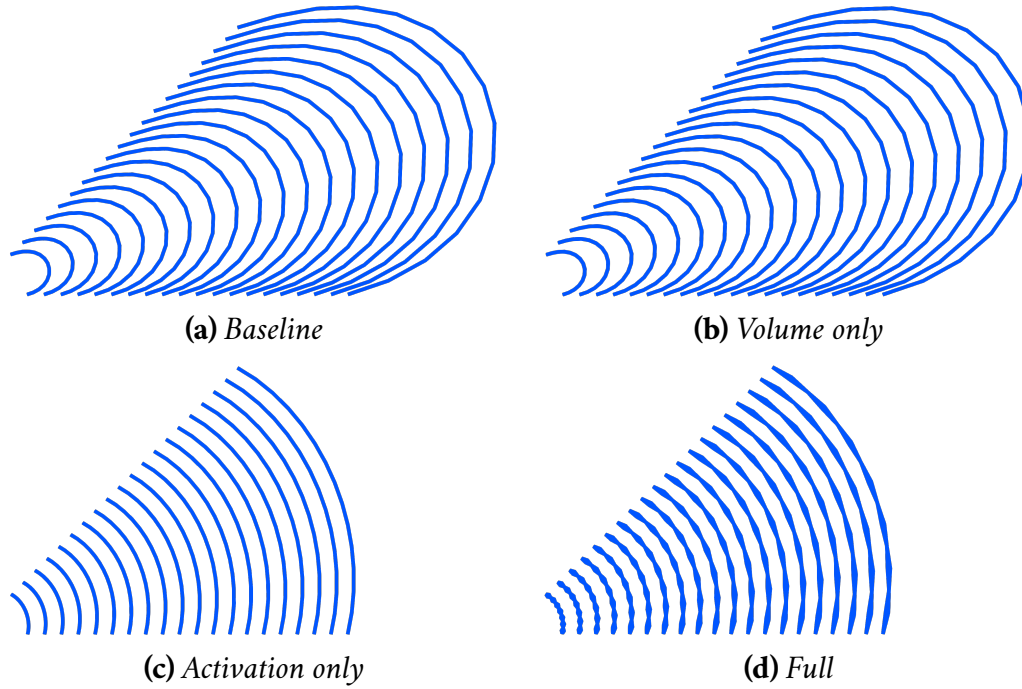


Figure 8.7: ParaView visualization of the arc experiment at the final deformation step. The passive configurations (a and b) exhibit pronounced bulging due to the lack of activation-driven contraction, while the active configurations (c and d) preserve the arc shape better

8.2.3 RQ3: Volume preservation

The third research question addresses the central contribution of this thesis, whether the volume constraint preserves segment volumes during deformation or not. This is evaluated by tracking the volume ratio $V(t)/V(0)$ of each rod during the simulation, where $V(t)$ is the total rod volume at step t and $V(0)$ is the initial rest volume.

Metric. The volume ratio $V(t)/V(0)$ tracks each rod's volume change over the course of the simulation. A ratio of 1.0 indicates perfect volume preservation. The volume constraint appears to be inactive without activation as the stretch-shear constraint alone maintains edge lengths close to their rest values, so the volumes remain near the initial value. Only when activation shortens the rest lengths does the volume

constraint respond by increasing the radii to compensate for the shortened segments. That is the core volume preservation mechanism described in Section 6.2.3.

Results. Figure 8.8 shows the volume ratio evolution. The volume-only and full configurations remain at 1.0 throughout, confirming perfect volume preservation in both passive and active scenarios. The baseline configuration stabilizes around 0.95, a 5 % volume loss as the stretch-shear constraint alone cannot fully maintain rest lengths, although higher solver iterations may help. The activation-only configuration decreases linearly from 1.0 to approximately 0.52, mirroring the linear progression of the contraction ratio κ over the 100 simulation steps. This confirms both that the contraction mechanism works correctly and that the volume constraint is necessary to preserve volume under activation-driven contraction.

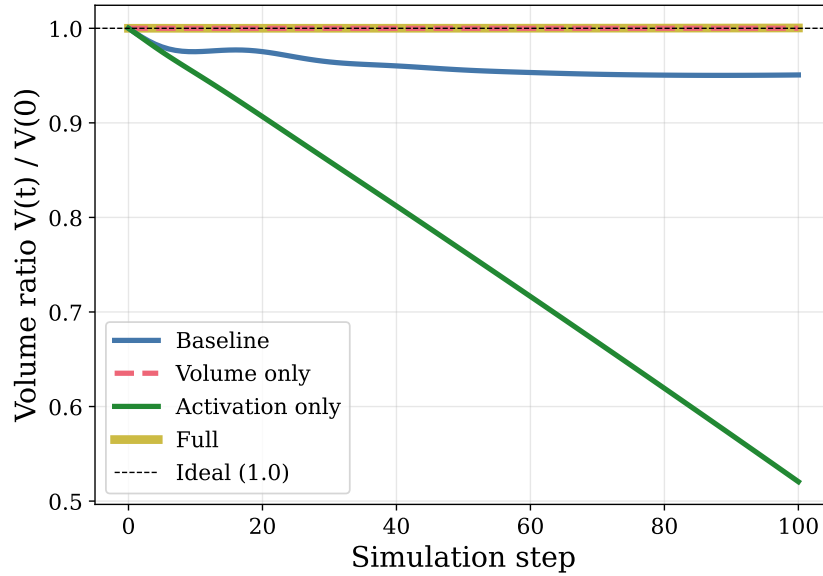


Figure 8.8: Mean volume ratio $V(t)/V(0)$ across all rods over the course of the simulation, per configuration

The effect of volume preservation on the radii is visualized in Figure 8.9. The baseline (a) and activation-only configurations (c) show uniform radii at the rest value of 0.005 units. The volume-only configuration (b) adjusts the radii marginally, though the difference is not visually discernible. In the full configuration (d), the volume constraint increases the radii to approximately 0.006–0.009, visible as bright yellow colors in the visualization. The alternating color pattern is a beading artifact caused by adjacent segments competing over shared vertex radii, discussed further in Chapter 9.

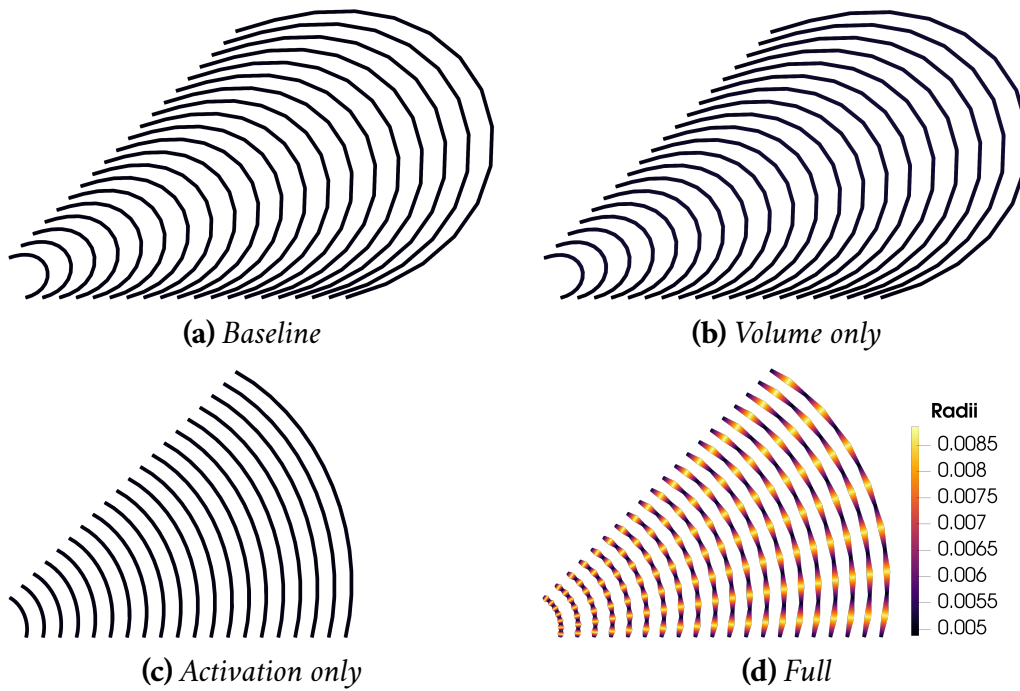


Figure 8.9: Radius visualization at the final deformation step, colored by vertex radius, scale [0.005, 0.009]

8.2.4 Summary of arc experiment results

The quarter-circle arc experiment answers all three research questions:

1. **RQ1 (Uniformity):** Rod vertices remain well-distributed across all configurations, confirming that Cosserat rods are a suitable representation for fiber deformation.
2. **RQ2 (Shape preservation):** Activation is necessary for shape preservation. Without activation, the rods retain their original rest lengths.
3. **RQ3 (Volume preservation):** The volume constraint successfully preserves volume in all scenarios. Without the volume constraint, activation-driven contraction causes significant volume loss.

The three results form a progression: rods maintain plausible geometry (RQ1), activation is required for correct shape (RQ2), and the volume constraint preserves volume during activation by adjusting radii (RQ3). The volume constraint provides no visible benefit without activation, but becomes essential when contraction shortens rest lengths. This validates the design decision to couple volume preservation with activation-driven contraction.

8.3 Hip flexion experiment

The quarter-circle arc experiment validated the rod constraint system in a controlled setting. This section evaluates whether those results transfer to a realistic musculoskeletal scenario: hip flexion driven by OpenSim activation data within the full Muscle Wrapping 2.0 pipeline.

8.3.1 Setup

The experiment simulates hip flexion from 0° to 90° over 50 simulation steps using the same four muscles as Havlíček [Hav24]: gluteus maximus, gluteus medius, iliacus, and adductor brevis. Bone kinematics and muscle activations are obtained from OpenSim’s SOT, which provides per-muscle activation signals at each simulation step. The rod-based simulation uses 100 fibers per muscle with a line resolution of 15 (i.e., 14 segments and 15 vertices per fiber), generated by the Hoang decomposition method [Hoa23]. The solver runs 20 iterations per time step with slice distance constraints configured for $K = 6$ nearest and $K = 2$ furthest neighbors. The remaining solver parameters match those in Table 8.1. The contraction ratio is set to $\kappa = 0.4$ – the mean value of the physical shortening estimate discussed in Section 6.4. A passive configuration with $\kappa = 0.0$ (no activation) is included for comparison.

8.3.2 Visual comparison: constraint contributions

Figure 8.10 shows the four muscles during hip flexion from 0° to 90° under three configurations, mirroring the progression from the arc experiment (RQ1–RQ3) on physiological muscle driven by OpenSim activation data. The muscles are colored as follows: gluteus maximus (green), gluteus medius (blue), iliacus (yellow), and adductor brevis (red). The baseline configuration (top row) shows the effect of the stretch-shear and bend-twist constraints alone, without activation or volume preservation – defacto representing the VIPER approach [Ang+19].

The effect of activation, or rather the lack thereof, is most apparent on iliacus (Figure 8.10b), which visibly bulges outward compared to the other configurations, which maintain a more compact shape due to the contraction mechanism. The volume constraint’s effect is more subtle: in Figure 8.10f, the iliacus fibers appear slightly thicker, while those of gluteus maximus are slightly thinner due to the stretching, compared to the passive and activation-only configurations.

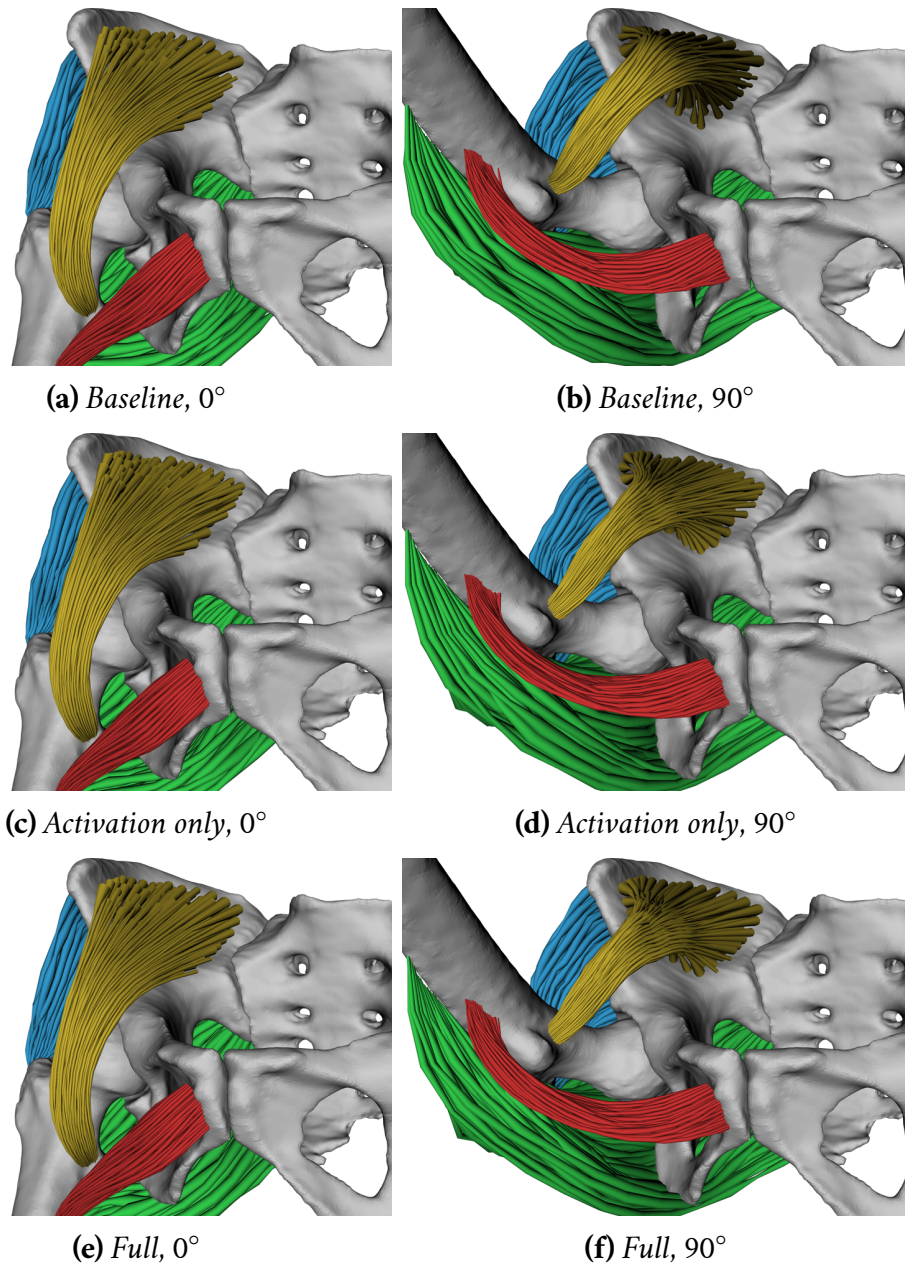


Figure 8.10: Ventral view of hip flexion at 0° (left) and 90° (right) under three different configurations: baseline (top, $\kappa = 0.0$ without volume preservation), activation only (middle, $\kappa = 0.4$ without volume preservation), and full (bottom, $\kappa = 0.4$ with volume preservation)

The progression from baseline to full configuration mirrors the arc experiment findings: activation prevents outward buckling, and the volume constraint adds the radial compensation that produces physiologically plausible muscle shapes.

8.3.3 Visual comparison: rod-based vs. surface-based

The rod-based and surface-based approaches use fundamentally different geometric representations – tube-rendered polyline versus triangulated meshes – so a direct visual comparison does not evaluate the same features. Nevertheless, since both methods simulate the same muscles under the same activation data and bone kinematics, comparing the resulting shapes is informative. If two distinct approaches produce similar results, it increases confidence in both methods.

A comparison of both methods from the ventral view at 0° and 90° is shown in Figure 8.11. The solver parameters are identical across both methods except two: the iteration count per time step, set to 20 for the rod-based approach and 70 for the surface-based one, as used by Havlíček [Hav24], and the fiber resolution, set to 15 per fiber for the rod-based approach and 100 for the surface-based one. Each method uses the configuration under which it performs best – lower fiber resolution produced unreliable contraction on the surface mesh, while higher resolution degraded decomposition quality and solver performance for the rod-based approach.

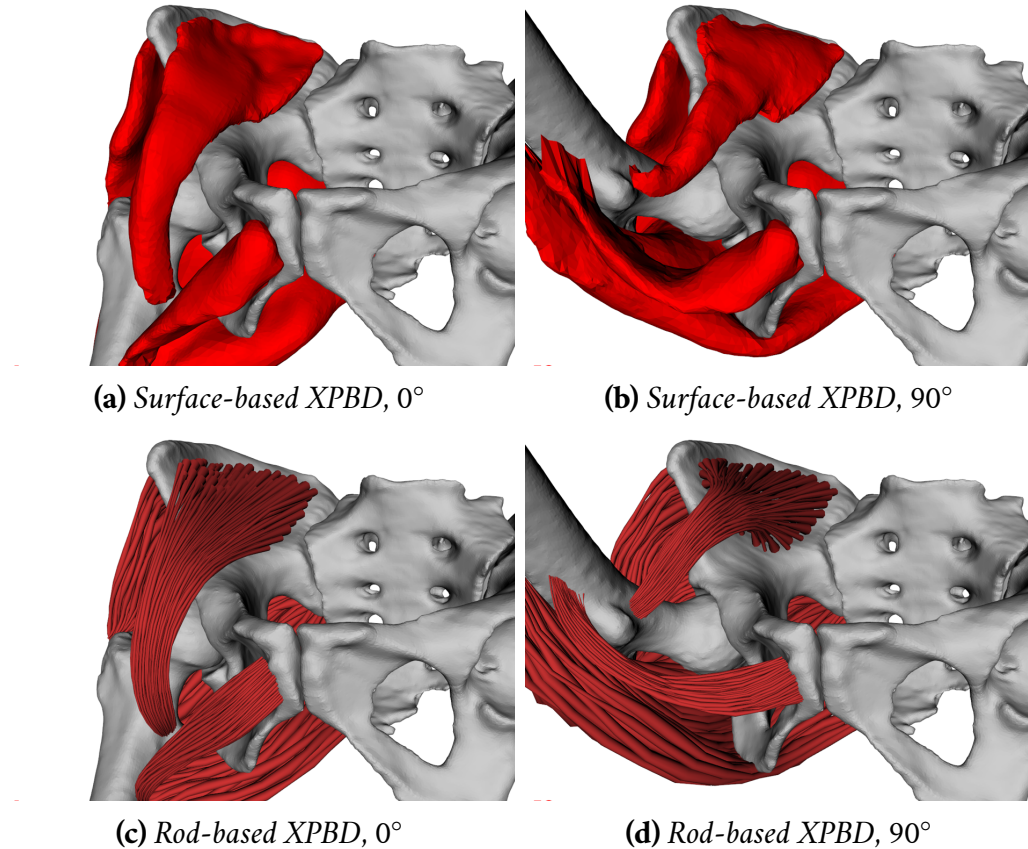


Figure 8.11: Ventral view of hip flexion at 0° (left) and 90° (right). Top row: Havlíček's surface-based XPBD implementation [Hav24]. Bottom row: rod-based XPBD approach

At 0° , both methods produce similar shapes for all four muscles. At 90° , the overall shapes remain comparable, but the rod-based approach exhibits noticeably less sagging in the iliocus and adductor brevis. This is expected, as the rod-based method contracts internal fibers directly via the stretch-shear constraint, which is closer to the physiological contraction mechanism. In contrast, the surface-based approach contracts along the surface fibers and reconstructs internal fiber behavior from the deformed mesh – an indirect approach that also faces greater mechanical resistance when propagating contraction through a triangulated mesh with thousands of vertices and competing constraints.

8.3.4 Fiber length comparison

Beyond visual plausibility, the rod-based approach must produce fiber lengths that follow expected biomechanical trends. Figure 8.12 compares fiber lengths between Havlíček's surface-based XPBD and the proposed rod-based method during hip flexion from -10° to 90° .

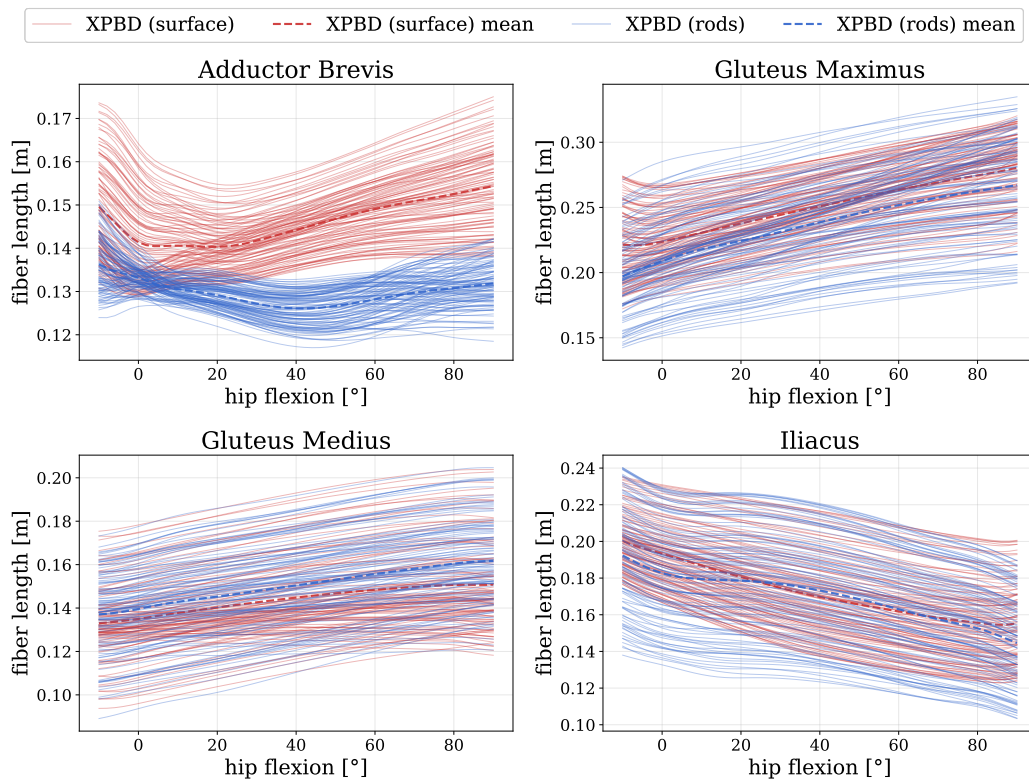


Figure 8.12: Fiber length comparison between surface-based (red) and rod-based (blue) methods during hip flexion. Thin lines represent individual fibers, the thick dashed line shows the mean

Both methods produce similar overall trends for all four muscles. The iliocus

shortens throughout flexion, consistent with its role as a hip flexor. The gluteus maximus and gluteus medius exhibit lengthening, which is expected for hip extensors and abductors during flexion. The adductor brevis shows a more complex pattern, with an initial shortening followed by lengthening, in both methods.

The rod-based fibers are, with the exception of gluteus medius, consistently shorter than the surface-based ones, with iliocus and adductor brevis showing the largest differences in the final stages of flexion. This behavior is expected with the sagging observed in the surface-based method for these two muscles (Section 8.3.3).

The overall shorter fibers in the rod-based method stem from the faster contraction propagation through the rod-based representation, as stated in Section 8.3.3. To verify that this behavior is not specific to hip flexion, Figure 8.13 shows the same comparison during hip abduction from 0° to 40° .

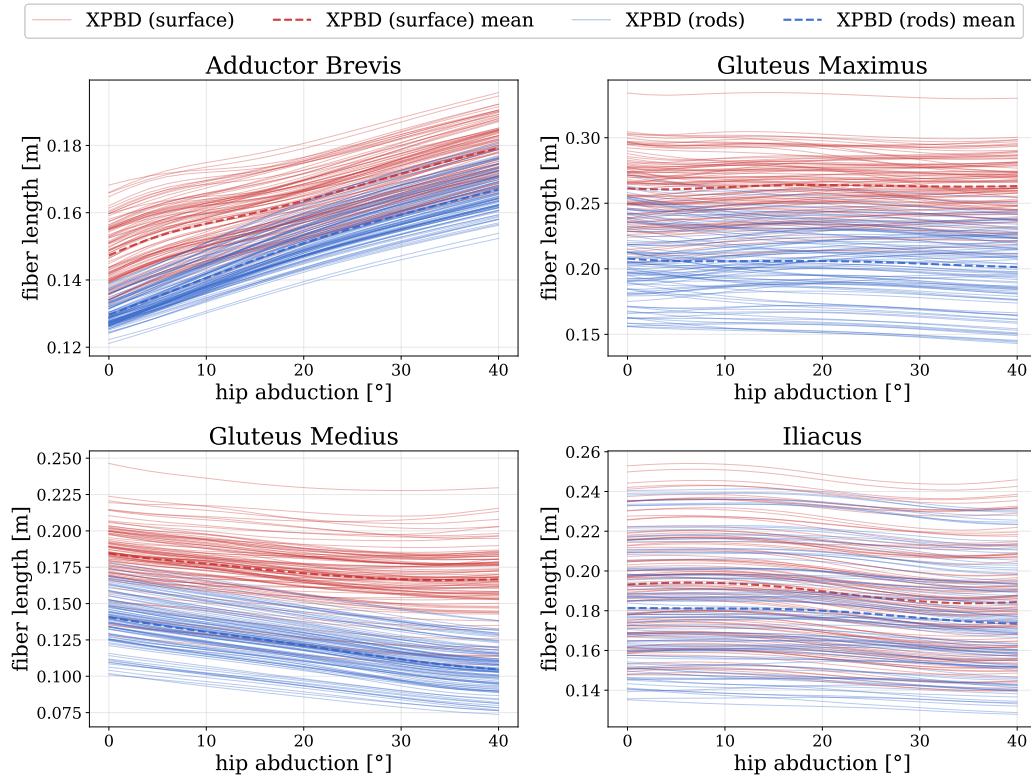


Figure 8.13: Fiber length comparison between surface-based (red) and rod-based (blue) methods during hip abduction. Thin lines represent individual fibers, the thick dashed line shows the mean

During hip abduction, the rod-based fibers remain consistently shorter across all muscles, confirming that the offset is a systematic difference between the two methods rather than a specific behavior during hip flexion.

8.3.5 Moment arm comparison

Moment arms quantify each muscle fiber's leverage around a joint, and serve as a key biomechanical metric for assessing the physiological plausibility of the simulated muscle shapes. The moment arm $r_{i,j}$ of the i -th fiber with respect to the j -th coordinate is computed using the tendon excursion method [MK20]:

$$r_{i,j} = \frac{\partial l_i}{\partial \theta_j},$$

where l_i is the fiber length and θ is the joint angle. The simulation computes the moment arms for all fibers at each time step during the movement. For gluteus maximus and gluteus medius (hip extensors), their moment arms are directly reported as hip extension moment arms (positive when the muscles lengthen during flexion). For the iliacus, a hip flexor, the sign is negated so that positive values indicate hip flexion moment arms, consistent with the literature data used for comparison.

To assess physiological plausibility, the rod-based method's moment arms are compared against digitized¹ data from Blemker & Delp [BD05] and Modenese & Kohout [MK20], both of which provide moment arm bands for discretized fiber representations of the iliacus, gluteus maximus, and gluteus medius, as well as Kumazaki et al. [KUM+22], which provides data for iliacus, albeit not as a band. The literature data are overlaid as shaded regions or data points in Figure 8.14 and Figure 8.15. No literature comparison is available for adductor brevis.

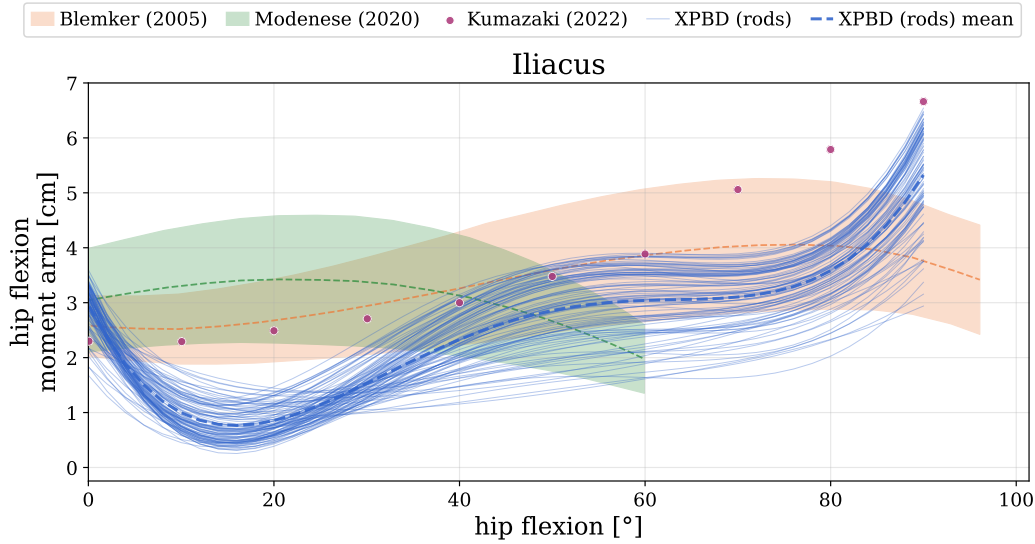


Figure 8.14: Iliacus hip flexion moment arms during hip flexion. Rods are represented as blue lines. Shaded regions show the bands from Blemker [BD05] (orange) and Modenese [MK20] (green). Data points from Kumazaki et al. [KUM+22] are shown in magenta. The colored dashed lines show the respective means

¹ Digitized using WebPlotDigitizer at: <https://apps.automeris.io/wpd4/>

For the iliacus, the rod-based moment arms range from 1–2 cm in the mid-range (15° – 40°) and rise to 2–4 cm beyond 60° . The rod-based band sits below both the Blemker [BD05] and Modenese [MK20] bands in the mid-range, but converges toward the Blemker region at higher flexion angles, following the same upward trend. The Kumazaki et al. data points [KUM+22], measured from anatomical specimens, follow the same upward trend and converge with the rod-based values at higher flexion angles. The steep spike at negative hip flexion angles is a numerical artifact at the boundary of the motion range and should be disregarded.

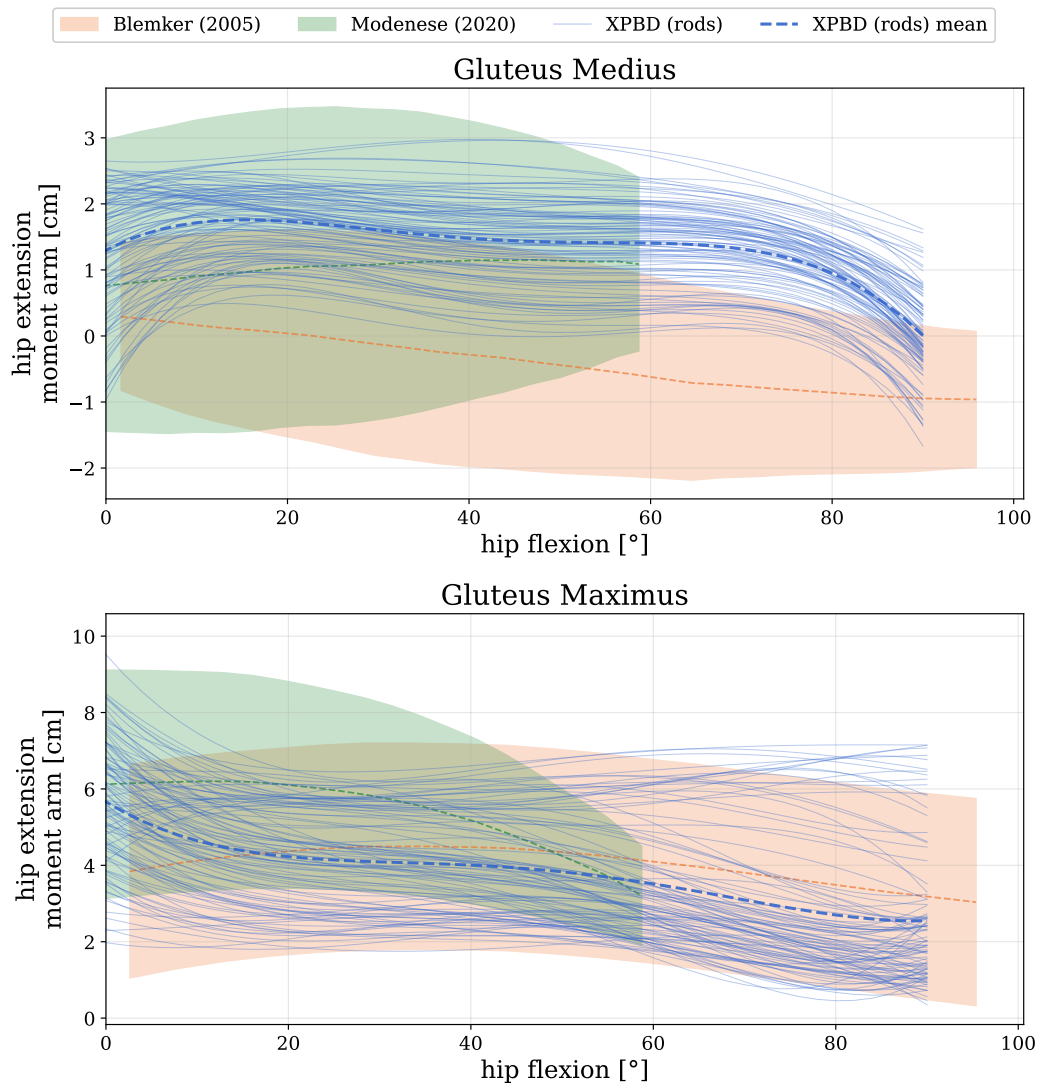


Figure 8.15: *Gluteus medius (top) and gluteus maximus (bottom) hip extension moment arms during hip flexion. Legend is the same as in Figure 8.14*

The gluteus medius moment arms fall almost entirely within the Modenese band and partially overlap with the Blemker band. The transition from positive to

negative moment arms around 80° can be explained as the muscle changing roles from hip extensor to hip flexor at high flexion angles.

For gluteus maximus, the rod-based moment arms overlap extensively with both literature bands across the entire motion range, showing the strongest agreement among the three muscles.

To quantify the agreement, Table 8.2 shows two metrics adapted from Modenese et al. [MK20]. The *envelope overlap* measures the percentage of the literature envelope area that is covered by the rod-based simulation envelope (i.e., the intersection area divided by the literature area). The *fiber containment* measures, at each step, the percentage of rod-based fibers whose moment arms fall within the literature range, averaged across all steps. These two metrics capture both the spatial coverage (does the simulation cover the literature range?) and the distributional agreement (do the simulated fibers land inside the reference band?).

Table 8.2: *Moment arm agreement metrics between the rod-based simulation and literature data^{*}. The range column indicates the movement interval over which the metrics are computed*

Muscle	Reference	Envelope	Containment	Range
iliacus	Blemker 2005	46.1 %	46.7 %	-6.9° – 90.0°
iliacus	Modenese 2020	37.9 %	34.1 %	-8.2° – 59.9°
glut. medius	Blemker 2005	37.6 %	31.9 %	1.7° – 90.0°
glut. medius	Modenese 2020	60.5 %	99.3 %	-8.6° – 58.7°
glut. maximus	Blemker 2005	91.3 %	95.2 %	2.6° – 90.0°
glut. maximus	Modenese 2020	74.4 %	73.5 %	-8.9° – 58.7°

^{*} Since Kumazaki reports discrete data points rather than a band, no envelope or containment metrics are computed for this reference

For the iliacus, both references show moderate agreement (46 %/47 % against Blemker, 38 %/34 % against Modenese). The low values come from the mid-range underestimate visible in Figure 8.14: the rod-based moment arms cluster around 1–2 cm in the 15° – 40° range, while the literature bands sit at 2–4.5 cm. However, the rod-based moment arms follow the same trend as Blemker, which could be explained by the rod-based contraction shortening the fibers more effectively (as shown in Section 8.3.4), reducing the length change during flexion and thus the moment arm.

Gluteus medius agreement is split: the Modenese band, which is broader, achieves 99 % containment, while the Blemker band captures only 32 % of fibers. The alignment with Modenese can be explained by the usage of the same bone model geometry and moment arm computation method.

Gluteus maximus shows the strongest agreement, with over 91 % overlap and 95 % containment against Blemker, and around 74 % for both metrics against Modenesi. This is consistent with the visual overlap in Figure 8.15, and raises confidence in the physiological plausibility of this muscle during hip flexion.

Overall, the rod-based moment arms capture the general trends for all three muscles and fall within or near the literature for most of the motion range, with the Kumazaki data confirming the upward trend for iliacus at higher flexion angles. The mid-range underestimate for iliacus and the wider spread for gluteus maximus at higher flexion angles suggest that moment arm accuracy is sensitive to fiber routing and collision handling near bone surfaces. Nevertheless, the level of agreement (achieved without any moment-arm specific tuning) supports the conclusion that the Cosserat rod representation produces biomechanically plausible results.

8.4 Performance analysis

A key practical advantage of the rod-based approach is its computational efficiency. Table 8.3 compares per-frame solver timing² between three configurations: Havlíček’s surface-based XPBD implementation [Hav24], the rod-based approach with and without parallel constraint projection. Both methods use shared solver parameters (Table 8.1) on the same hardware (see Section 8.1), each in its best configuration – 100 vertices per fiber and 70 solver iterations per frame for the surface-based method, and 15 vertices and 20 iterations for the rod-based method. Parallelization is controlled by a single compile-time flag (`ENABLE_ROD_PARALLEL_PROCESSING`), which switches per-rod operations between sequential and parallel.

Table 8.3: *Per-frame timing of the surface-based and rod-based approaches*

Approach	Frame time	Frame rate	Dominant costs
surface-based	~ 900 ms	~ 1 FPS	constraints ~ 564 ms joint activations ~ 140 ms bone collision ~ 133 ms
rod-based	~ 248 ms	~ 4 FPS	bone collision ~ 226 ms slice constraint ~ 8 ms rod constraints ~ 14 ms
rod-based (parallel)	~ 38 ms	~ 26 FPS	bone collision ~ 26 ms slice constraint ~ 9 ms rod constraints ~ 3 ms

² Frame time measures only the XPBD solver loop (`XPBD::execute()`). The reported FPS additionally includes visualization, VTK polydata export, and other overheads, and is therefore lower than the raw solver performance.

Without parallelization, the rod-based approach is already approximately $4\times$ faster than the surface-based one. Parallelization provides an additional $6.5\times$ speedup, bringing the total improvement to around $24\times$ over the surface-based. This speedup is possible due to the *embarrassingly parallel* nature of rod constraints. Each rod's stretch-shear, bend-twist, and volume constraints operate exclusively on that rod's vertices and quaternions, sharing no degrees of freedom with other rods. This allows for straightforward parallelization across rods without synchronization. In contrast, Havlíček's surface constraints operate on a shared mesh topology with triangles sharing vertices, so parallelization required a greedy graph coloring algorithm to identify independent constraint groups – a significantly more complex approach that still could not fully eliminate sequential bottlenecks.

Rod-bone collision dominates the parallel frame budget, accounting for around 68 % of the total frame time (26 ms out of 38 ms). This operation samples each rod edge at 6.5 mm intervals and queries the bone SDF for each sample. Further performance improvements should therefore focus on optimizing the collision detection (e.g., spatial hashing, bounding volume hierarchies) rather than the constraint projection, which already accounts for only 3 ms per frame. Slice distance constraints (9 ms) are currently sequential because they couple vertices across rods, but constraints on different slices do not share vertices, making this a candidate for future parallelization.

The debug visualizer (`vtkMAFVisualDebugger`) introduces a progressive memory leak that causes FPS to degrade over the course of the simulation. This is a known issue in the visualization library and does not affect the solver timing reported in Table 8.3.

Summary

The experiments in this chapter demonstrate that the proposed volume invariant Cosserat rod method produces physiologically plausible muscle deformation in both controlled (arc) and realistic (hip flexion) scenarios. The arc experiment validated the individual constraint contributions – activation drives contraction, the volume constraint preserves volume under contraction, and the combination produces the expected bulging. The hip flexion experiment confirmed that these properties transfer to a realistic musculoskeletal simulation, producing fiber lengths and moment arms that align with the surface-based approach and with published literature data [BD05; MK20]. The rod-based approach achieves this while running approximately $24\times$ faster than the surface-based method. The following chapter discusses the limitations of these results and identifies directions for future work.

Limitations and future work

9

The current implementation has several limitations that suggest avenues for future development:

1. **Beading artifact:** Adjacent volume constraint segments compete over shared vertex radii, producing alternating large-small radius pattern along the rod. A red-black Gauss-Seidel solver (i.e., processing odd and even segments in separate passes) would decouple these interactions.
2. **No muscle-muscle collision:** Currently, rods from different muscles can interpenetrate freely, though this was not observed during the hip flexion experiment with the 4 muscles. SDF-based or proximity-based collision handling (e.g., *virtual edges* from Havlíček [Hav24]) between rod bundles would be needed.
3. **Collision detection cost:** Rod-bone SDF queries dominate the frame time at 68 %, driven by the adaptive edge-level sampling required to catch mid-edge bone penetration. A straightforward optimization would be to restrict SDF collision checks to the outermost (shell) rods of each muscle bundle, since interior rods are shielded from bone contact by the outer rods.
4. **Fixed contraction ratio:** The contraction ratio κ is a global constant, while real muscles exhibit spatially varying contraction – the muscle belly contracts the most, while the tendinous attachment regions exhibit near zero contraction. Per-fiber or per-segment κ values would better capture this physiological behavior.
5. **Validation scope:** Only hip flexion, abduction, and adduction on a single anatomical model were tested. Extending the validation to other joints (e.g., knee), multi joint motions, and multiple anatomical models would strengthen confidence in the method's generality.

6. **Parallelization opportunities:** Slice distance constraints on independent slices do not share vertices and could be solved in parallel, reducing the current 9 ms sequential solve time.
7. **Code maintainability:** Splitting `XPBD.cxx` into two source files – one for Havlíček’s surface methods and one for the rod methods introduced in this thesis – would improve long-term maintainability of the codebase.

Conclusion

10

This thesis extended the Muscle Wrapping 2.0 XPBD-based muscle deformation pipeline with volume-invariant Cosserat rods, addressing the fundamental limitation of surface-based methods: their inability to capture internal fiber behavior and volume preservation. The rod-based representation was integrated directly into Havlíček's solver loop as a third entity class alongside muscles and bones, reusing Červenka's SDF collision and Havlíček's activation pipeline without modification.

Despite the limitations identified in the previous chapter, the proposed method successfully demonstrates that rod-based muscle simulation with volume preservation is both feasible and performant within the Muscle Wrapping 2.0 framework. Based on these results, the goals of the thesis can be considered fulfilled.

Bibliography

- [Ang+19] ANGLES, Baptiste et al. VIPER. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. 2019, vol. 2, no. 2, pp. 1–26. Available from DOI: 10.1145/3340260.
- [Arn+09] ARNOLD, Edith M.; WARD, Samuel R.; LIEBER, Richard L.; DELP, Scott L. A Model of the Lower Limb for Analysis of Human Movement. *Annals of Biomedical Engineering*. 2009, vol. 38, no. 2, pp. 269–279. ISSN 1573-9686. Available from DOI: 10.1007/s10439-009-9852-5.
- [BMM17] BENDER, Jan; MÜLLER, Matthias; MACKLIN, Miles. *A Survey on Position Based Dynamics*. The Eurographics Association, 2017. Available from DOI: 10.2312/EGT.20171034.
- [Bet+13] BETTS, J. Gordon et al. Anatomy and Physiology. In: OpenStax, 2013, chap. 11-1. Available also from: <https://openstax.org/books/anatomy-and-physiology/pages/11-1-interactions-of-skeletal-muscles-their-fascicle-arrangement-and-their-lever-systems>.
- [BD05] BLEMKER, Silvia; DELP, Scott. Three-Dimensional Representation of Complex Muscle Architectures and Geometries. *Annals of Biomedical Engineering*. 2005, vol. 33, no. 5, pp. 661–673. Available from DOI: 10.1007/s10439-005-1433-7.
- [Čer19] ČERVENKA, Martin. *Deformace svalových vláken systémem vázaných částic*. Západočeská univerzita v Plzni, 2019. Available also from: <http://hdl.handle.net/11025/37413>.
- [CC09] COSSERAT, E.; COSSERAT, F. Théorie des Corps déformables. *Nature*. 1909, vol. 81, no. 2072, pp. 67–67. Available from DOI: 10.1038/081067a0.
- [FTS08] FUNCK, Wolfram von; THEISEL, Holger; SEIDEL, Hans-Peter. Volume-preserving Mesh Skinning. In: *International Symposium on Vision, Modeling, and Visualization*. 2008. Available also from: <https://api.semanticscholar.org/CorpusID:10048745>.

- [Gas+23] GASH, Matthew C.; KANDLE, Patricia F.; MURRAY, Ian V.; VARACALLO, Matthew A. *Physiology, Muscle Contraction*. StatPearls, 2023. Available also from: <https://www.ncbi.nlm.nih.gov/books/NBK537140/>.
- [Gaz+18] GAZZOLA, M; DUDTE, L; MCCORMICK, A; MAHADEVAN, L. Forward and inverse problems in the mechanics of soft filaments. *Royal Society Open Science*. 2018, vol. 5, no. 6, p. 171628. Available from doi: 10.1098/rsos.171628.
- [Gre10] GREEN, Simon. *Particle Simulation using CUDA*. 2010. Whitepaper. NVIDIA Corporation. Available also from: <https://developer.download.nvidia.com/assets/cuda/files/particles.pdf>.
- [Hav24] HAVLÍČEK, Ondřej. *Muscle interaction in the context of muscle deformation modelling by a Position Based Dynamics method*. Západočeská univerzita v Plzni, 2024. Available also from: <http://hdl.handle.net/11025/57230>.
- [Hoa23] HOANG, Duc Long. *Automatic Generation of Internal Fibres of Skeletal Muscles*. Západočeská univerzita v Plzni, 2023. Available also from: <http://hdl.handle.net/11025/53764>.
- [Hsu+25] HSU, Jerry; WANG, Tongtong; WU, Kui; YUKSEL, Cem. Stable Cosserat Rods. In: *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. ACM, 2025, pp. 1–10. Available from doi: 10.1145/3721238.3730618.
- [KK14] KOHOUT, J.; KUKAČKA, M. Real-Time Modelling of Fibrous Muscle. *Computer Graphics Forum*. 2014, vol. 33, no. 8, pp. 1–15. Available from doi: 10.1111/cgf.12354.
- [KS16] KUGELSTADT, Tassilo; SCHÖMER, Elmar. *Position and Orientation Based Cosserat Rods*. The Eurographics Association, 2016. Available from doi: 10.2312/SCA.20161234.
- [KUM+22] KUMAZAKI, TOSHIMASA; TAKAHASHI, TOMIHISA; NAKANO, TAKASHI; SAKAI, TATSUO. Action and Contribution of the Iliopsoas and Rectus Femoris as Hip Flexor Agonists Examined with Anatomical Analysis. *Juntendo Medical Journal*. 2022, vol. 68, no. 4, pp. 352–362. ISSN 2188-2126. Available from doi: 10.14789/jmj.jmj22-0009-0a.
- [MMC16] MACKLIN, Miles; MÜLLER, Matthias; CHENTANEZ, Nuttapong. XPBD: position-based simulation of compliant constrained dynamics. In: *Proceedings of the 9th International Conference on Motion in Games*. ACM, 2016. MiG '16. Available from doi: 10.1145/2994258.2994272.

- [Mar+20] MARTIN, Meg; TRAVOUILLO, Kenny; FLEMING, Patricia; WARBURTON, Natalie. Review of the methods used for calculating physiological cross-sectional area (PCSA) for ecological questions. *Journal of Morphology*. 2020, vol. 281, no. 7, pp. 778–789. Available from doi: 10.1002/jmor.21139.
- [MMG06] MERRY, Bruce; MARAIS, Patrick; GAIN, James. Animation space: A truly linear framework for character animation. *ACM Trans. Graph.* 2006, vol. 25, pp. 1400–1423.
- [MK20] MODENESE, Luca; KOHOUT, Josef. Automated Generation of Three-Dimensional Complex Muscle Geometries for Use in Personalised Musculoskeletal Models. *Annals of Biomedical Engineering*. 2020, vol. 48, no. 6, pp. 1793–1804. Available from doi: 10.1007/s10439-020-02490-4.
- [MLH20] MOO, Eng; LEONARD, Timothy; HERZOG, Walter. The sarcomere force-length relationship in an intact muscle-tendon unit. *Journal of Experimental Biology*. 2020. Available from doi: 10.1242/jeb.215020.
- [Mül+07] MÜLLER, Matthias; HEIDELBERGER, Bruno; HENNIX, Marcus; RATCLIFF, John. Position based dynamics. *Journal of Visual Communication and Image Representation*. 2007, vol. 18, no. 2, pp. 109–118. ISSN 1047-3203. Available from doi: 10.1016/j.jvcir.2007.01.005.
- [Pai02] PAI, Dinesh. STRANDS: Interactive Simulation of Thin Solids using Cosserat Models. *Computer Graphics Forum*. 2002, vol. 21, no. 3, pp. 347–352. Available from doi: 10.1111/1467-8659.00594.
- [RD13] RIEBL, Shaun; DAVY, Brenda. The Hydration Equation. *ACSM'S Health & Fitness Journal*. 2013, vol. 17, no. 6, pp. 21–28. Available from doi: 10.1249/fit.0b013e3182a9570f.
- [Sai+24] SAILLANT, Bastien; ZARA, Florence; DAMIAND, Guillaume; JAILLET, Fabrice. High-order elements in position-based dynamics. *The Visual Computer*. 2024, vol. 40, no. 7, pp. 4737–4749. Available from doi: 10.1007/s00371-024-03467-3.
- [SSD13] SHERMAN, Michael; SETH, Ajay; DELP, Scott. What is a Moment Arm? Calculating Muscle Effectiveness in Biomechanical Models Using Generalized Coordinates. In: *Volume 7B: 9th International Conference on Multibody Systems, Nonlinear Dynamics, and Control*. American Society of Mechanical Engineers, 2013. Available from doi: 10.1115/detc2013-13633.

- [Tak+13] TAKAZA, Michael; MOERMAN, Kevin; GINDRE, Juliette; LYONS, Garry; SIMMS, Ciaran. The anisotropic mechanical behaviour of passive skeletal muscle tissue subjected to large tensile strain. *Journal of the Mechanical Behavior of Biomedical Materials*. 2013, vol. 17, pp. 209–220. Available from DOI: 10.1016/j.jmbbm.2012.09.001.
- [USS14] UMETANI, Nobuyuki; SCHMIDT, Ryan; STAM, Jos. Position-based elastic rods. In: *ACM SIGGRAPH 2014 Talks*. ACM, 2014, pp. 1–1. Available from DOI: 10.1145/2614106.2614158.
- [Wik06] WIKIMEDIA COMMONS. *File:Cylinder.svg* [online]. 2006. [visited on 2025-12-28]. Available from: <https://commons.wikimedia.org/w/index.php?curid=1154925>.
- [Wik07] WIKIMEDIA COMMONS. *File:Fiederung.svg* [online]. 2007. [visited on 2025-12-30]. Available from: <https://en.wikipedia.org/wiki/File:Fiederung.svg>.
- [Wik08] WIKIMEDIA COMMONS. *File:CroppedCone.svg* [online]. 2008. [visited on 2025-12-28]. Available from: <https://commons.wikimedia.org/w/index.php?curid=4284775>.
- [Wik23] WIKIMEDIA COMMONS. *File:The Sliding Filament Theory Visual Model.pdf* [online]. 2023. [visited on 2026-01-02]. Available from: https://commons.wikimedia.org/wiki/File:The_Sliding_Filament_Theory_Visual_Model.pdf.
- [Wik24] WIKIMEDIA COMMONS. *File:PoissonRatio.svg* [online]. 2024. [visited on 2026-01-02]. Available from: <https://commons.wikimedia.org/wiki/File:PoissonRatio.svg>.
- [Zha+19] ZHANG, Xiaotian; CHAN, Fan; PARTHASARATHY, Tejaswin; GAZZOLA, Mattia. Modeling and simulation of complex dynamic musculoskeletal architectures. *Nature Communications*. 2019, vol. 10, no. 1. Available from DOI: 10.1038/s41467-019-12759-5.

List of Figures

2.1	The hierarchical structure of skeletal muscle, from the whole muscle down to the sarcomere	8
2.2	Common fascicle arrangements in skeletal muscle	9
2.3	Predefined fiber templates: parallel (left) and pennate (right)	10
2.4	Pennate muscle fiber arrangements. PCSA in green, ACSA in blue	11
2.5	Gluteus medius (left) and iliacus (right) with their attachment areas. Blue landmarks identify the origin area, while the red landmarks correspond to the insertion area	12
2.6	The sliding filament mechanism showing cross-bridge formation and power stroke	13
2.7	Types of muscle contraction. Concentric (top), eccentric (middle) and isometric (bottom)	14
2.8	Muscle roles during elbow flexion	15
2.9	Illustration of the Poisson effect before (green) and after (red) an object is stretched. The load in the longitudinal axis x elicits a response in both the lateral axes, y and z	16
2.10	Subfigures (a) and (c) show the collapsing joint and candy wrapper effects when an object bends and twists respectively. Subfigures (b) and (d) are free of these phenomena	17
3.1	Cosserat rod with the triad $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ representing the Eulerian frame while the vectors $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$ describe the local reference frame	22
3.2	Comparison between classical Cosserat rods (left) and VIPER rods (right) under the effects of stretching and compression	24
3.3	Discretization of the centerline into a series of connected line segments. Vertices in red, edges in dark blue	25
3.4	Discretization of a continuous Cosserat rod into segments with an associated local reference frame $\mathbf{R}_i$ for orientation and an edge defined by two vertices $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$	26
3.5	Reference frames represented as unit quaternions $\mathbf{q}_i$ at each segment of the discretized rod	26

3.6	From left to right: Illustration of the pectoral, biceps and deltoid muscles and the representation of their radii at vertices	29
3.7	A truncated cone (frustum) with radii r_1 and r_2 at its ends and height h	29
3.8	A cylinder with diameter d and height h	30
5.1	Architecture of Muscle Wrapping 2.0	40
5.2	Gluteus medius deformation over 1, 100, and 200 simulation steps using Červenka's PBD method. Please note the form preservation issues at the 200-th simulation step surrounding the attachment area	42
5.3	Perpendicularly running detailed surface fibers on the iliacus muscle	44
5.4	Kukačka method for muscle decomposition. A – program input, B – harmonic scalar field of the muscle surface, C – extraction of iso-lines, D – fiber template selection, E – merge the sliced muscle and template, F – program output	46
5.5	Viperization method for muscle decomposition	47
5.6	Hoang method for muscle decomposition. A – program input, B – harmonic scalar field of the muscle surface, C – extraction of iso-lines, D – point sampling on each slice, E – Munkres algorithm	47
6.1	Simplified illustration of the small-world network principle applied to slice distance constraints. Without long range connections (a), information from node A requires 6 hops to reach node B and 5 to reach C. Adding a single furthest-neighbor connection (b) reduces $A \rightarrow B$ to 1 hop and $A \rightarrow C$ to 2 hops via the shortcut A-B-C	63
7.1	Effect of the overflow factor on radius assignment	72
7.2	Cosserat rod initialization process	74
7.3	Volume constraint effect on iliacus (yellow) during hip flexion	76
7.4	Slice distance constraint effect on gluteus maximus during hip flexion	76
7.5	Gluteus maximus penetrating the pelvis without edge-level collision handling	78
7.6	Effect of EMA smoothing and rate clamping on raw activation signals from OpenSim over a 250 frame motion sequence. Left: raw (light blue) versus smoothed (dark blue) activation curves. Right: corresponding frame-to-frame deltas showing that the smoothed signal stays well within the maximum delta threshold, while the raw signal exhibits sharp spikes that manifest as visible twitching in the simulation	80

7.7	Arc deformation experiment, initial state (left) and deformed state (right). The origin $[0,0]$ serves as the joint center, while the x and y axes represent the bones. Fixed rod endpoints are anchored along the axes and are driven by the bone transforms to produce the desired angular deformation	84
8.1	Vertex distribution along a deformed arc rod. Uniform spacing (left) produces well-behaved segments. Collapsed vertices (right) create short degenerate segments near the fixed endpoints and long overstretched segments elsewhere	88
8.2	Distribution of normalized segment-length ratios at the final deformation step across all rods, per configuration. The dashed line marks the ideal value of 1.0, while CV is the mean coefficient of variation across all rods	89
8.3	Polar error decomposition. Point sim_1 (blue) shows a radial error ε_r – correct angular position but wrong distance from the origin, while point sim_2 (red) shows an angular error ε_θ – correct radius but offset along the arc	90
8.4	Simulated (colored) vs. analytical arc (black) positions for rod 10 at the final deformation step, per configuration. The passive configurations (baseline and volume-only) show large outward deviation because the rod retains its original rest length, which is too long for the shorter 45° arc. The active configurations (activation-only and full) shorten the rest length which keeps the rod close to the reference arc	91
8.5	Per-vertex radial (left) and angular (right) errors at the final deformation step for rod 10. The passive and active configurations show a stark difference. In the radial deviation plot, the passive configurations (blue and pink) show visibly more bulging than the active configurations (green and yellow), the ideal is zero error (dashed line). In the angular deviation, the passive configurations show negative errors for the first half of the rod and positive errors from vertex 7 onward	92
8.6	Fréchet distance across all 20 rods per configuration	92
8.7	ParaView visualization of the arc experiment at the final deformation step. The passive configurations (a and b) exhibit pronounced bulging due to the lack of activation-driven contraction, while the active configurations (c and d) preserve the arc shape better	93
8.8	Mean volume ratio $V(t)/V(0)$ across all rods over the course of the simulation, per configuration	94
8.9	Radius visualization at the final deformation step, colored by vertex radius, scale $[0.005, 0.009]$	95

8.10	Ventral view of hip flexion at 0° (left) and 90° (right) under three different configurations: baseline (top, $\kappa = 0.0$ without volume preservation), activation only (middle, $\kappa = 0.4$ without volume preservation), and full (bottom, $\kappa = 0.4$ with volume preservation)	97
8.11	Ventral view of hip flexion at 0° (left) and 90° (right). Top row: Havlíček's surface-based XPBD implementation [Hav24]. Bottom row: rod-based XPBD approach	98
8.12	Fiber length comparison between surface-based (red) and rod-based (blue) methods during hip flexion. Thin lines represent individual fibers, the thick dashed line shows the mean	99
8.13	Fiber length comparison between surface-based (red) and rod-based (blue) methods during hip abduction. Thin lines represent individual fibers, the thick dashed line shows the mean	100
8.14	Iliacus hip flexion moment arms during hip flexion. Rods are represented as blue lines. Shaded regions show the bands from Blemker [BD05] (orange) and Modenese [MK20] (green). Data points from Kumazaki et al. [KUM+22] are shown in magenta. The colored dashed lines show the respective means	101
8.15	Gluteus medius (top) and gluteus maximus (bottom) hip extension moment arms during hip flexion. Legend is the same as in Figure 8.14 . .	102

List of Tables

5.1	Comparison of relevant approaches for muscle decomposition	48
5.2	Comparison of existing approaches for muscle simulation	50
7.1	Source files added or modified by this thesis	70
8.1	Solver parameters shared across experiments	87
8.2	Moment arm agreement metrics between the rod-based simulation and literature data*. The range column indicates the movement interval over which the metrics are computed	103
8.3	Per-frame timing of the surface-based and rod-based approaches . . .	104

List of Listings

7.1	Abridged CosseratRod class header	73
7.2	Compile-time feature flags and their usage	82
7.3	<code>mw2::for_each()</code> and usage	83

1101001 1100001
10101100001110010 1100001
101011010101 1100001



11010011101101001
011000011010101
110000101011101