EXTENDING PROJECTIVE DYNAMICS FOR PRACTICAL AND EFFICIENT SIMULATION

by

Qisi Wang

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2025

Date of final oral examination: 05/09/25

The dissertation is approved by the following members of the Final Oral Committee:
Eftychios Sifakis, Professor, Computer Science
Mohit Gupta, Associate Professor, Computer Science
Josiah Hanna, Assistant Professor, Computer Science
Dan Negrut, Professor, Mechanical Engineering

To my parents, for their unwavering support and belief in me.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Eftychios Sifakis, for his invaluable mentorship, guidance, and encouragement throughout my doctoral studies. His intellectual rigor, high standards, and unwavering support have profoundly shaped my approach to research and problem solving. Throughout this journey, I have continued to be in awe of the breadth of his knowledge, from mechanical formulation and mathematical solvers to performance optimization and software engineering. His example will always remind me to stay curious and to dig deep into every problem I face.

I will never forget the many hours we spent writing code together in his office, and the whiteboards covered with equations and derivations, each session a lesson in both technical mastery and thoughtful mentorship. I am especially grateful for the freedom he gave me to explore my ideas, the clarity he brought to my thinking, and the confidence he instilled in me during uncertain times. From him, I have learned not only how to pursue meaningful research, but also how to care for people and support others with empathy and integrity. Most of all, I am deeply grateful that he never gave up on me. This dissertation would not have been possible without his insight, generosity, and commitment.

I would like to thank the other members of my committee, Professor Mohit Gupta, Professor Josiah Hanna, and Professor Dan Negrut, for their support and engagement with my work throughout my doctoral studies. I am especially grateful for the thoughtful questions and valuable perspectives they shared during my defense, which challenged me to reflect more deeply on my contributions and their broader impact. Their encouragement and insight have been truly meaningful and have played an important role in the completion of this dissertation.

I would like to express my heartfelt thanks to my collaborator, Dr. Court Cutting, whose deep passion for plastic surgery, perseverance, and eagerness to learn about programming and simulation concepts have been truly inspiring. He approaches everything he does with integrity, and holds a sincere belief in the value of our work—not only as a scientific contribution but also as a way to help others. His

dedication and authenticity have been a constant source of motivation for me. I have learned so much from him—not just professionally, but also about what it means to be a good person. Our time working together has left a lasting impression and will continue to inspire me to become a more thoughtful, caring, and responsible member of society.

From him, I have learned the importance of lifelong learning and contributing meaningfully at any stage of life. He has taught me to reflect deeply on my career and to always keep in mind its potential to help others. I am especially grateful for the time that he took to meet with me, to understand the computer science side of our project, and to offer insightful and intuitive explanations of the clinical background and application. His trust, patience, and collaborative spirit have meant a great deal to me, and I truly value everything we have built together.

I am also sincerely thankful to my colleagues and other collaborators, including Yutian Tao, Sangeetha Grama Srinivasan, and Eric Brandt, for their many thoughtful discussions, technical contributions, and camaraderie. Their insights and feedback continuously challenged me to think more deeply and pushed this work to greater clarity and precision. It was a privilege to learn and grow alongside them.

Special thanks go to our program coordinator, Angela Thorp, for helping me navigate the many complicated administrative processes along the way. I deeply appreciate her patience, detailed guidance, and the inspiring encouragement she offered—especially during the final stages of my PhD journey, when her support meant so much.

I would also like to thank my dear friends, Professor Robert Krainer and his wife Lynne Krainer, who have become like grandparents to me from another country. I feel incredibly grateful and lucky to have met them. From them, I have felt truly loved and cared for. I deeply appreciate the way they welcomed me into their lives—introducing me to new experiences, encouraging me to meet new people, and, most importantly, teaching me through their example how to love and care for others with kindness and generosity.

Special thanks go to my friends Ting Huang, Siyu Lu, and Xiaomin Zhang, whose support, humor, and steady presence helped carry me through some of the

most challenging moments of this journey. Their encouragement—both in and outside of academia—has meant more to me than words can express.

Finally, I am profoundly grateful to my family and my partner for their unconditional love, patience, and belief in me. Their quiet sacrifices, constant encouragement, and enduring support gave me the strength to keep moving forward, even when the path was unclear. This milestone is as much theirs as it is mine.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This dissertation presents a series of extensions to the Projective Dynamics (PD) framework for simulating elastic soft bodies. PD is widely used for its speed and robustness, enabled by a constant system matrix that allows fast direct solves and parallel local steps. However, this strength also imposes limitations when simulating more complex behaviors that require changing system matrices.

We investigate how to retain the efficiency and robustness of PD while enabling support for

1. collision handling
2. nonlinear tissue behavior
3. self actuation simulation
4. differentiable simulation for integration with deep learning frameworks

For collision, we exploit the locality of contact forces and use a Schur Complement-based decomposition to preserve most of the factorized system. For simulating skin and tissue, we introduce a strain-limiting material model that fits within the PD paradigm. For self-actuation, we adopt Shape Targeting and show how it can be simulated efficiently with only minor changes to the PD local step. For differentiable simulation, we propose an iterative correction scheme that reuses the PD global matrix and avoids full Hessian solves in the backward pass.

These techniques are integrated into a high-performance simulator that supports interactive rates even with hundreds of thousands of tetrahedral elements. We demonstrate applications including a facial flap surgery simulator with self-contact and surgical constraints, and a differentiable physics layer embedded in a neural network for learning muscle actuation. Together, these results extend the range of physical behaviors that can be simulated within the PD framework, while preserving its original advantages.

PREFACE

Simulating elastic soft bodies in computer graphics presents a trade-off between speed, stability, and physical realism. A good interactive‘ simulation method must be fast enough for interactive use, robust enough to handle extreme deformations without numerical instability, and flexible enough to support a wide range of materials and behaviors.

Many existing simulation frameworks work by repeatedly solving large systems of equations—often with over 100,000 variables—to compute how objects should move. These systems become even more complex when modeling realistic effects like rotation, collision, or nonlinear material behavior. In such settings, achieving both efficiency and robustness is notoriously difficult.

Projective Dynamics (PD) offers a compelling solution. It splits each simulation step into two parts: a fast, parallelizable local update and a global solve that uses a constant matrix. This structure enables large-scale simulations to run quickly and stably. However, PD's speed depends on strong assumptions about material behavior and simulation conditions. In practice, many scenarios—like tissue deformation, muscle actuation, or contact with other objects—violate these assumptions.

This dissertation expands the Projective Dynamics framework to handle such challenging scenarios without sacrificing its speed or stability. In particular, I explore how to adapt PD for:

- **Collisions:** Making PD-compatible with contact and self-contact events without slowing down the solver.
- **Nonlinear tissue:** Supporting materials that stiffen when stretched—important for simulating biological tissues.
- **Self-actuation:** Enabling active deformations like muscle contractions within PD's efficient structure.
- **Differentiable simulation:** Making PD work as a layer in deep learning models by enabling gradient computation through the simulation.

Each of these extensions is designed to preserve PD's key advantage—a constant system matrix that can be prefactored for speed—while broadening its applicability to a much wider class of simulation problems.

Note on Co-authorship

This dissertation is based on collaborative research conducted during my PhD. While all chapters involve contributions from co-authors, I led the majority of the implementation, experimentation, and writing related to the simulation systems and methods. Each chapter includes a co-authorship note outlining the collaborators and my specific contributions.

1.1 Motivation

Projective Dynamics (PD) [Bouaziz et al. (2014)] is a widely adopted framework for interactive simulation of deformable objects, particularly those governed by corotated elasticity [McAdams et al. (2011)].

It can be viewed as a quasi-Newton method that employs a constant approximate Hessian, enabling stable and efficient simulation via an alternating two-phase scheme, while ultimately converging to the same solution as a traditional Newton-Raphson method. Each time step is formulated as an energy minimization problem, solved via an alternating local-global scheme: per-element projections followed by a constant-matrix global solve. The algorithm guarantees quasi-monotonic energy descent in each substep, offering robustness even under large deformations and allowing for early termination of iterations without compromising stability.

PD's efficiency stems from the constancy of its global system matrix: a Laplacian-like structure that is independent of the object's configuration and can be prefactored once, enabling fast direct solves via forward and backward substitution.

These properties make PD particularly well-suited for interactive soft-body simulation, where user-driven inputs may cause extreme deformations, element inversions, or unpredictable dynamics that demand both numerical robustness and fast response times. In such contexts, early termination of time integration steps is often necessary to maintain real-time performance, and PD's guaranteed stability under incomplete convergence becomes a critical advantage. Moreover, the ability to handle large deformations while preserving rotational invariance ensures that PD can deliver visually plausible results even under challenging simulation conditions.

In contrast, full-Newton methods achieve high physical fidelity by directly solving the first-order optimality conditions of nonlinear energy functionals. However, their system matrices—the true Hessians of the elastic energy—are configuration-dependent and typically indefinite, particularly under large or inverted deforma-

tions. As a result, direct solvers require repeated matrix factorizations at every iteration, while iterative solvers demand sophisticated preconditioners to ensure convergence. These demands make full-Newton methods prohibitively expensive for high-resolution or real-time simulation. Furthermore, their stability is not guaranteed without auxiliary techniques such as line searches or trust-region methods—especially when solver iterations are truncated for speed, a common necessity in interactive applications.

1.2 Challenges

PD's computational efficiency hinges on a key structural assumption: that the global system matrix remains constant across time steps. This structural constancy enables fast prefactorization but limits compatibility with many real-world phenomena. For instance, collision handling introduces constraints that change dynamically as contacts form and break. Nonlinear tissue responses—such as strain stiffening in biological materials—violate the quadratic energy assumption. Internal actuation, like muscle contractions, involves deformation-dependent forces. Finally, integrating PD into learning frameworks requires differentiating through the solver—a task made difficult by the assumption of a constant system matrix. Each of these scenarios challenges PD's original design and motivates the extensions developed in this dissertation.

Collision

Collision handling introduces configuration-dependent forces that violate the key structural assumption of Projective Dynamics: a constant global system matrix. In typical soft-body simulations, contact is resolved through penalty-based methods—such as zero-rest-length springs or barrier potentials—that act only when surface points interpenetrate. These forces depend on the current geometry of the mesh and dynamically modify the simulation energy functional as contacts form, evolve, or disappear.

In PD, the global step solves a prefactored Laplacian-like matrix derived from fixed topology and material models. Directly including the contact terms, however, alters the sparsity and values of this matrix, invalidating the prefactorization and requiring expensive re-factorization at each time step. This undermines PD's computational efficiency and stability guarantees.

In contrast, full-Newton methods handle such changes seamlessly. Their system matrix—the true Hessian of the energy—is already recomputed each iteration, naturally incorporating any new contact terms. Matrix-free solvers like Conjugate Gradient are particularly well suited here, as they avoid explicit factorization altogether.

Nonlinear Tissue Response

Biological materials such as skin and soft tissue often display highly nonlinear elastic behavior—particularly a biphasic response, where stiffness increases rapidly beyond a certain strain threshold. This kind of response is critical to capture in applications like surgical simulation, where the realism and predictive power of the model depend on accurate deformation under high strain.

Traditional Newton-Raphson methods handle such nonlinearities naturally: by directly solving the first-order optimality conditions of nonlinear energy functionals—such as Mooney-Rivlin or Ogden models—through iterative linearization. The system matrix (the true Hessian of the energy) is updated at each iteration, capturing the local curvature of the energy landscape regardless of its complexity.

In contrast, PD relies on elastic energies that are quadratic in form and independent of the current deformation state, enabling a fixed system matrix and efficient local-global updates. Incorporating nonlinear materials into this framework breaks the assumption of a constant stiffness matrix, thereby invalidating the prefactorization that underpins PD's efficiency. As a result, simulating nonlinear strain-stiffening behavior remains a major challenge within the PD paradigm.

Internal Actuation

Many soft-body systems in graphics and biomechanics involve internal actuation—such as muscle-driven motion in faces or limbs—where deformation arises not only from external forces but also from control-dependent internal stimuli. Prior constitutive models [Teran et al. (2003, 2005b)] achieve actuation by introducing directional constraints that contract or expand in response to control inputs.

These approaches effectively make the material's rest configuration a function of the control signal, altering the local energy landscape and, critically, changing the structure of the global system matrix. This deformation-dependent variability violates the core assumption in PD that the system matrix remains constant, thereby invalidating the prefactorization strategy and limiting PD's ability to simulate active materials without structural changes to its solver.

Differentiability

Recent advances in physics-informed learning have driven the need for differentiable simulators—frameworks capable of computing gradients of simulation outputs with respect to control signals, material parameters, or other inputs. These gradients are essential for training models via backpropagation in applications such as soft robotics, inverse design, and learned actuation for character animation.

In Projective Dynamics, however, the global step uses a constant prefactored system matrix that is independent of the current deformation. Computing true gradients typically requires solving a linear system involving the actual Hessian of the energy, which varies with the simulation state. This discrepancy makes it nontrivial to extract correct gradients while preserving the computational benefits of PD's constant-matrix structure.

1.3 Contributions

This dissertation presents a series of enhancements to Projective Dynamics (PD), addressing its core limitations through scalable algorithms, physically informed

models, and practical system design. The contributions span three technical chapters, each corresponding to a key axis of extension: collision handling, nonlinear tissue behavior, and internal actuation with differentiability.

Chapter 4: Core Simulator and Collision Handling

- **High-Performance PD-Based Simulator:** We develop an interactive, high-resolution simulator for volumetric elastic bodies using the PD paradigm. The simulator supports real-time performance on tetrahedral meshes with over half a million elements. Local steps are optimized using SIMD-parallel projection routines, while the global step leverages the constancy of the system matrix to enable prefactorization and fast direct solves via libraries such as Intel® MKL PARDISO.

- **Efficient Collision Resolution via Partial Factorization:** We introduce a method for incorporating collision response into PD without disrupting the global system structure. When the collision-prone surface region is sparse and known in advance, we isolate it using a partial Cholesky factorization and solve the resulting Schur complement efficiently on the GPU. A selective local step update is also applied to only the collision-affected region, preserving global efficiency while enabling responsive contact dynamics.

Chapter 5: Nonlinear Tissue Behavior and Clinical Application

- **Strain-Limiting Extension for Nonlinear Tissue Response:** To capture biphasic responses observed in biological tissues such as skin, we introduce a high-stiffness penalty energy that activates under large strain. This term is integrated into the local projection step, enabling stiffness modulation while keeping the constancy of the global system matrix. The approach maintains PD's efficiency and is well suited for simulating biological soft tissues.

- **Facial Flap Simulator with Surgical Tool Constraints:** We apply our enhanced PD framework to develop a cognitive simulator for facial flap pro-

cedures. Custom PD-compatible constraints model common surgical tools, while real-time self-contact handling enables accurate simulation of flap undermining and closure. Compared to Newton-style solvers, PD supports interactive frame rates with robust nonlinear elasticity, enabling practical deployment in clinical training tools.

Chapter 6: Internal Actuation and Differentiability

- **Efficient Actuation via Shape Targeting:** We extend PD to simulate internally actuated deformable objects using shape-targeting energies. These energies—similar in form to corotated elasticity—are seamlessly integrated into the local-global iteration scheme, enabling efficient simulation of active materials such as muscles while preserving matrix constancy.

- **Differentiable PD via Hybrid Adjoint Solver:** To support learning and inverse problems, we developed a differentiable PD variant that enables gradient computation with respect to actuation inputs. We adopt the adjoint method and solve the required linear system via a hybrid approach: alternating between matrix-free residual computations using the true Hessian and correction steps with the constant PD matrix. This approach enables efficient gradient computation without sacrificing simulation performance.

- **Integration into Deep Learning Pipelines:** We incorporate our differentiable PD solver as a physics-based decoder within neural network architectures. This allows for learning control signals that produce desired shapes through physically consistent simulation. The system is scalable to meshes with over 250,000 elements and supports end-to-end training for high-fidelity actuation models.

Computational Foundations and Related Work

To situate the technical contributions of this dissertation within the broader context of deformable simulation, Chapter 2 provides a focused literature review on the core methodologies underpinning this work—namely, corotated elasticity and Projective Dynamics. Chapter 3 then lays out the technical background, detailing how PD achieves its efficiency and robustness. These two components form the computational foundation of the simulator developed in this thesis. Additional literature related to specific extensions—such as collision handling, nonlinear constitutive models, and differentiable simulation—will be reviewed in the respective chapters where those contributions are presented.

2.1 Corotated Elasticity

Simulation of deformable bodies using corotated elasticity strikes a good balance between respecting nonlinearity and rotational invariance while revealing opportunities for interactive simulation. The principle of Corotated Elasticity first materialized in warped stiffness methods [Müller et al. (2002)], and later made rotationally invariant [Müller and Gross (2004)], and robust to inversion [Irving et al. (2004)] and indefiniteness of the stiffness matrix [Teran et al. (2005b)]. Analytic second derivatives of the corotated energy allowed improved convergence of Newton Methods [McAdams et al. (2011); Chao et al. (2010)] while the derivative singularity of the model around highly compressed configurations was treated with appropriate modifications [Stomakhin et al. (2012)].

2.2 Projective Dynamics

Proposed by [Bouaziz et al. (2014)], Projective Dynamics draws its inspiration from Position Based Dynamics [Müller et al. (2006); Bender et al. (2013); Müller et al. (2014)], interpreting elastic models as collections of constraint. However, by deriving the method from a continuum mechanics point of view, unlike Position Based Dynamics, where simulation results depend on the resolution of simulation mesh and timestep size, material stiffness can be directly specified in Projective Dynamics.

Due to its efficiency and robustness, Projective Dynamics has since then become a popular time integration method for interactive simulation. Chebyshev iteration has also been used to tackle the global step [Wang (2015); Wang and Yang (2016)], allowing efficient GPU implementation, albeit carrying weaker guarantees for robustness relative to direct solvers. Additionally, graph coloring is utilized for a GPU implementation of Projective Dynamics in [Fratarcangeli et al. (2016)]. [Peng et al. (2018)] accelerate the convergence of Projective Dynamics by applying

Anderson acceleration. On the other hand, [Brandt et al. (2018)] proposed a hyper-reduction method for the constraint space to reduce computation cost per PD iteration.

Throughout the years, Projective Dynamics has enjoyed significant adoption and evolution, dealing with various simulation scenarios. It has been used for developing damping models [Li et al. (2018)], elastic rod simulations [Soler et al. (2018)], face animation [Ichim et al. (2017)], motion control using volumetric actuators [Lee et al. (2018a)]. [Komaritzan and Botsch (2018)] applies Projective Dynamics for character skinning with kinematically controlled bones. Their follow-up work [Komaritzan and Botsch (2019)] further accelerates their collision handling with a GPU-implemented preconditioned CG method as the linear solver for the global matrix. Projective Dynamics has also been extended to support two-way couplings of deformable and rigid material for articulated character simulation in [Li et al. (2019)]. [Ly et al. (2020)] introduced the inclusion of the Signorini-Coulomb law in Projective Dynamics and handles nodal contacts with dry friction. [Li et al. (2021)] proposed incremental updates to the Cholesky factor of the global matrix to deal with the low-rank changes resulting from tearing and cutting.

As a backward Euler integrator for dynamics systems, Projective Dynamics suffers from numerical damping intrinsic to such a method. To deal with the damping, [Dinev et al. (2018b)] applies a similar global-local alternating scheme for implicit-midpoint integration to reduce numerical damping and enforce energy conservation of the simulation result through blending with the result of the forward- or backward- Euler scheme. Co-currently, [Dinev et al. (2018a)] devise methods to amortize the numerical damping intrinsic to the implicit Euler integration scheme by projecting the solution to an energy conservation manifold in a post-processing procedure. Furthermore, focusing on the same issue, [Kee et al. (2021)] devises physically based energy and momentum preservation constraints to further counter angular momentum loss in Projective Dynamics.

On the theory side, [Liu et al. (2017)] cast Projective Dynamics as a quasi-newton method and extend its applicability to more general hyper-elastic constitutive models by utilizing the global matrix of Projective Dynamics as an initial approximation

of the hessian matrix in an L-BFGS scheme. On the other hand, [Narain et al. (2016); Overby et al. (2017)] realize that Projective Dynamics with affine constraints manifold can be viewed as a particular case of alternating direction method of multipliers, allowing more general constitutive models and constraints to be used, with iterative solvers utilized for the global step, albeit typically demonstrated at more modest resolutions than we use.

3 TECHNICAL BACKGROUND

3.1 Notation

We start by establishing some notational conventions to assist with clarity of our exposition. In this document, we denote variables that represent aggregate quantities (e.g. concatenated lists of a physical property on *all nodes*, or *all elements*) by using boldface type – *e.g.*all mesh vertices $\vec{\mathbf{x}} := (\vec{x}_1, \ldots, \vec{x}_n)$. These aggregate quantities can be either scalar or vector, which are differentiated by an arrow over the variable for vector quantities. For example $\mathbf{y}$ might be the y-coordinates of all vertices in a mesh, while $\vec{\mathbf{v}}$ might be all 3d vertices of the mesh, consisting of the three components $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}$. Subscripts in parenthesis denote iteration numbers, for example $\vec{\mathbf{x}}_{(2)}$ might be the 3D values of all the mesh vertices after the second iteration of an algorithm. Matrices are capital Roman letters (non-bolded). Aggregate matrices are bolded versions of the single matrix notation. Refer to Table 3.1 for a complete list[1].

3.2 Corotated Elasticity

The constitutive model of corotated elasticity defines the energy density $\Psi(F)$ as a function of the deformation gradient

$$\Psi_{ce}(F) = \mu \|F - R(F)\|_F^2 + \frac{\lambda}{2} tr^2(R^T(F)F - I) \tag{3.1}$$

$$= \mu \|S(F) - I\|_F^2 + \frac{\lambda}{2} tr^2(S(F) - I) \tag{3.2}$$

where $R(F)$ and $S(F)$ are the rotational and symmetric component, respectively, of the deformation gradient, F, given by the polar decomposition, $F = RS$. μ and λ are the Lamé coefficients, and $\|\cdot\|_F$ is the Frobenius norm. Note that here all the

[1] We acknowledge certain unavoidable duplications of names (e.g. Σ), where we prefer to retain historical convention and let context connote the intended meaning.

Example	Type	Example Usage
x_i, y_i, z_i	Scalar quantity	x-, y-, or z-coordinate of the i^{th} mesh vertex
$\vec{x}_i$	Vector quantity	3d-vector position of the i^{th} mesh vertex
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Aggregate scalar quantity	All of the x-, y-, or z-coordinates of all the mesh vertices
$\mathbf{x}^{(i)}$	Aggregate scalar quantity	All of the i^{th} coordinates of all of the mesh vertices (typically $i \in \{1, 2, 3\}$)
$\vec{\boldsymbol{x}}$	Aggregate vector quantity	All of the 3d positions of all the mesh vertices
$\vec{x}_{(k)}$	Iteration	k^{th} iteration of a 3d-vector
$\vec{\boldsymbol{x}}_{(k)}$	Aggregate iteration	k^{th} iteration of all of the mesh vertices
F_e	Matrix	Deformation gradient of the e^{th} element of the mesh
$\mathbf{F}$	Aggregate matrix	All deformation gradients of the elements of the mesh

Table 3.1: Notation used in this thesis

quantities R, F, S, and Ψ are functions of the deformed configuration x, i.e. $F = F(x)$, $\Psi(F) = \Psi(F(x))$, etc. Here we omit the x terms for the simplicity of notation.

To build some intuition on corotated elasticity, let us recognize that the term $\epsilon_{ce} = S - I = \frac{1}{2}(S + S^T) - I$ is the corotated strain. The corotated energy, therefore, has the form

$$\Psi(\epsilon) = \mu ||\epsilon||_F^2 + tr^2(\epsilon) \tag{3.3}$$

which is the general energy form of isotropic hyperelastic material with a linear stress-strain relationship.

If instead of the corotated strain metric, we take the Cauchy's infinitesimal strain

$\epsilon_{inf} = \frac{1}{2}(F + F^T) - I$, we get the familiar energy of linear elasticity:

$$\Psi_{le} = \mu||\epsilon_{inf}||_F^2 + \frac{\lambda}{2}tr^2(\epsilon_{inf}) \tag{3.4}$$

$$= \mu||\frac{1}{2}(F + F^T) - I||_F^2 + \frac{\lambda}{2}tr^2(\frac{1}{2}(F + F^T) - I) \tag{3.5}$$

The similarity between the corotated strain and Cauchy's infinitesimal strain gives an intuition about corotated elasticity – it gives what linear elasticity would have behaved if we factored out the pure rotation component of the local deformation. Corotated elasticity, therefore, inherits its robustness to large deformation and inversion from linear elasticity. Moreover, the inclusion of the rotation term $R(x)$ makes corotated elasticity rotationally invariant, *i.e.*for any rotation matrix R, $\Psi(RF) = \Psi(F)$. This gives corotated elasticity models the ability to handle significant rotation without inducing ghost force infesting linear elasticity.

However, this nonlinearity introduced by the rotation comes with the cost that more sophisticated solvers - such as the traditionally used Newton Raphson method and the more recently proposed Projective Dynamics - are needed for the simulation steps.

Traditionally, each simulation step can be cast as a minimization problem. For a simulation with quasistatic time integration steps, the total energy over the whole domain Ω at time step n with the current configuration $x_{(n)}$ and time step size h is

$$E = \int_\Omega \Psi_{ce} dx \tag{3.6}$$

And for a simulation with implicit time integration steps, the total energy is

$$E = \int_\Omega \Psi_{ce} + \frac{1}{h^2}\int_\Omega \rho^{\frac{1}{2}}|x - s_{(n)}|_2^2 dx \tag{3.7}$$

where $s_{(n)} = x_{(n)} + hv_{(n)} + \frac{h^2}{\rho}f_{ext}$, ρ is density, and the current velocity is $v_{(n)} = (x_{(n)} - x_{(n-1)})$. Note that the external force f_{ext} here does not depend on the

configuration x.

If solved through Newton Raphson method, in each Newton's iteration k, the following linear system of equation is solved:

$$\left.\frac{\partial^2 E}{\partial \vec{x}^2}\right|_{x_k} \delta x = -\left.\frac{\partial E}{\partial x}\right|_{x_k} \tag{3.8}$$

$$\Rightarrow K_k \delta \vec{x} = -f_k \tag{3.9}$$

The Hessian matrix K_k is commonly referred to as the system/stiffness matrix of the problem.

Here we emphasize that, due to the nonlinearity of the strain-stress relationship in corotated elasticity, the system matrix is a function of the configuration of the current iteration x_k. Therefore the system matrix varies as Newton's iterations progress. For a direct solver, such as Cholesky factorization in conjunction with forward- and backward-substitution, this volatility in the system matrix will require the factorization stage of the solver to be performed in each Newton's step. Considering the typical sizes of simulation degree of freedom in production environments (in the magnitude of 100 K), the computation cost induced from the refactorization of matrices of such size is unbearable for real-time applications.

Another problem that usually need to be addressed when Newton's method is used for minimization of the corotated energy, is that, not only is the energy non-quadratic, it is also non-convex. Such property poses significant problem for the convergence, even the stability, of the newtons method and calls for additional safeguading measures such as definiteness fixes and line search.

3.3 Projective Dynamics

We start by reviewing the mathematical formulation for Projective Dynamics [Bouaziz et al. (2014)], but with a perspective more aligned with the language of continuum mechanics (including concepts such as stress and force) rather than

a purely-optimization treatment. With a volumetric tetrahedral model and linear elements [Sifakis and Barbic (2012)], we can compute a deformation gradient, $F_e(\vec{x})$, which is constant in each element e and is a linear function of the deformed locations $\vec{x}$ of the mesh vertices.

In keeping with the typical mode of use of Projective Dynamics, we omit the λ term by setting this value to zero. This also helps yield an optimized implementation. The effect of this term will be discussed in Section 7.2. We draw attention to the important detail that R is a dependent function of F in the formulation of corotated elasticity. We also note that we allow the option for the coefficient μ to either be a universal constant, or varies across the mesh, taking different values from one element to the next (while being constant within each element in a tetrahedral discretization), in which case we will denote its element-specific value by μ_e.

Projective Dynamics suggests an alternative formulation of Equation 3.2 where R is no longer a function of F, but rather an independent variable:

$$\hat{\Psi}(F, R) = \mu ||F - R||_F^2 \tag{3.10}$$

The fundamental observation at the core of Projective Dynamics is that the conventional description of the constitutive model's force density function, $\Psi(F)$, is equal to the minimum over all rotation matrices R of the Projective Dynamics energy density $\hat{\Psi}(F, R)$:

$$\Psi(F) = \min_{R \in SO(3)} \hat{\Psi}(F, R)$$

We transition from the (constant) energy density function across each element to an integrated energy function for the same element by multiplying by the (undeformed) volume of each element:

$$E_e(F_e) = \text{Vol}_e \Psi(F_e) = \min_{R_e \in SO(3)} \text{Vol}_e \hat{\Psi}(F_e, R_e) = \min_{R_e \in SO(3)} \hat{E}_e(F_e, R_e)$$

where we have defined $\hat{E}_e(F, R) := \text{Vol}_e \hat{\Psi}(F, R)$. The overall energy of the entire body (with rotations momentarily regarded as independent variables) is the sum

of all elemental energies:

$$\hat{E}(\vec{x}, \mathbf{R}) = \sum_e \hat{E}_e(F_e(\vec{x}), R_e) \tag{3.11}$$

from which we can recover the conventional discrete corotated energy for the entire mesh by minimizing rotations over all elements:

$$E(\vec{x}) = \min_{\mathbf{R}} \hat{E}(\vec{x}, \mathbf{R})$$

where it is implied (for brevity of notation) that the minimum is taken over an aggregate $\mathbf{R}$ of matrices that are all rotations (i.e. in $SO(3)$). We may intuitively interpret the energy $\hat{E}(\vec{x}, \mathbf{R})$ as a separate consitutive model from corotational elasticity, where each element's matrix R_e is no longer functionally tied to its deformation gradient, but is simply an element-specific simulation parameter, almost like μ_e.

Projective Dynamics is usable both in a quasistatic, as well as an implicit Backward Euler time integration scheme, as both cases are ultimately cast in very similar optimization problems. For simplicity of exposition, in this thesis we focus on the quasistatic case, with the understanding that our methodology remains fully applicable in the case where Backward Euler is used. In a quasistatic simulation, the deformable system evolves as to satisfy a force equilibrium condition, or equivalently in pursuit of a minimizer for the energy:

$$\min_{\vec{x}} E(\vec{x}) = \min_{\vec{x}, \mathbf{R}} \hat{E}(\vec{x}, \mathbf{R}) \tag{3.12}$$

Therefore, the quasistatic evolution can be seen as a minimization of the modified energy $\hat{E}$ jointly over both *independent* parameters $\vec{x}$ and $\mathbf{R}$. Projective Dynamics chooses to conduct this minimization by alternating the following two steps until convergence:

Local Step Treat $\vec{x}$ as constant, and minimize $\hat{E}$ over all $\mathbf{R}$.

Global Step Treat $\mathbf{R}$ as constant and minimize $\hat{E}$ over all $\vec{x}$.

The stability property of Projective Dyanmics results from the fact that each of these steps can be iterated while guaranteeing that the energy will monotonically decrease after the application of each one. The local step of minimizing $\mathbf{R}$ is actually multiple independent steps of minimizing R_e separately for *each* element. Because these R_e are independent, the problem is highly parallel. The solution to the minimization of R_e of each element is obtained via the Orthogonal Procrustes Problem and yields the minimizer $R_e = U_e(V_e)^T$ where $F_e = U_e\Sigma_e(V_e)^T$ is the SVD of F_e. The minimization associated with the *global step* will be handled by an application of a Newton-Raphson procedure, producing an iterative update sequence $\vec{x}_{(k+1)} \leftarrow \vec{x}_{(k)} + \delta\vec{x}$ where δx is computed by solving:

$$\left.\frac{\partial^2 \hat{E}}{\partial \vec{x}^2}\right|_{\vec{x}_{(k)}} \delta\vec{x} = -\left.\frac{\partial \hat{E}}{\partial \vec{x}}\right|_{\vec{x}_{(k)}} \tag{3.13}$$

On the left hand side, we recognize the second derivative of the reformulated energy from Equation (3.11). Having treated the rotations $\mathbf{R}$ as an independent parameter, this energy is a pure quadratic function of positions $\vec{x}$, thus the Hessian is a constant matrix. This also suggests that the Newton procedure in Equation (3.13) will terminate in just one iteration; we will retain this formulation though, in anticipation of emergence of further nonlinear (and non-quadratic) terms. When the expression in Equation (3.11) is interpreted as a modified constitutive model with $\mathbf{R}$ being an independent parameter, this Hessian would be intuitively associated with the "stiffness matrix" of this material model, which we denote as K_{el} (with the subscript denoting that this is the "elastic" energy, contrasted to collision-spawned contributions discussed later). Similarly on the right-hand side, we recognize the term $-\left.\frac{\partial \hat{E}}{\partial \vec{x}}\right|_{\vec{x}_{(k)}}$ as $\vec{f}_{el}(\vec{x}_{(k)})$, which are the aggregate elastic forces computed on the mesh nodes from this constitutive model, at position $\vec{x}_{(k)}$ [Sifakis and Barbic (2012)]. This allows us to equivalently write:

$$K_{el}\delta\vec{x} = \vec{f}_{el}(\vec{x}_{(k)}) \tag{3.14}$$

The stiffness matrix K_{el} can be computed either in the fashion of the Projective Dynamics formulation [Bouaziz et al. (2014)], or by following the Finite Element route which would produce exactly the same result. For the force $\vec{\mathbf{f}}_{el}(\vec{\mathbf{x}}_{(k)})$ however, we opt for a computation using the Finite Element paradigm: On each particular element, e, we start by calculating the deformation gradient, F_e, then calculating the first Piola stress tensor, $P(F_e)$, which in turn is used to calculate the force, $\vec{\mathbf{f}}_e$ [Sifakis and Barbic (2012)]. The first Piola stress tensor will be given by simply differentiating $\hat{E}(F_e) = \mu||F_e - R_e||_F^2$ with respect to F_e and recognizing that R_e is a constant:

$$P(F_e) = \frac{\partial \hat{E}}{\partial F_e} = 2\mu(F_e - R_e) \tag{3.15}$$

which is seemingly the same as that of corotated elasticity [McAdams et al. (2011)], with the caveat that R is still treated as an independent parameter rather than a function of F (the two will have been brought in sync, by virtue of the local step).

We find it advantageous to combine the computation of the forces $\vec{\mathbf{f}}_{el}$ with the execution of the local step, as the Piola stress of each element, and ultimately the element's contributions to $\vec{\mathbf{f}}_{el}$, are more efficiently computed at the same time that the local rotations are updated.

Having calculated $\vec{\mathbf{f}}_{el}$, we can obtain δx through Equation 3.13, and solving the resulting linear system by direct or indirect methods, such as a Cholesky factorization or conjugate gradient.

Thus, the difference between Newton-Raphson corotated elasticity, and the global step of Projective Dynamics corotated elasticity lies in the critical difference that the stiffness matrix on the left hand side of (Equation 3.13). In traditional corotated elasticity, this matrix incorporates all the nonlinearity of the full corotational model. In Projective Dynamics, the stiffness matrix is *constant*, independent of x_k, because given a fixed $\mathbf{R}$, $\hat{E}(\vec{\mathbf{x}}, \mathbf{R})$ is a pure quadratic in $\vec{\mathbf{x}}$ with a second-order coefficient that only depends on the simulation mesh's topology. Thus, an iteration using corototated Newton-Raphson is very similar to corotated Projective Dynamics, but the latter enjoys a much succinct stiffness matrix that is constant across all

global-local iterations. This analogy is detailed in [Liu et al. (2016)].

Furthermore, It can be shown that the global step stiffness matrix K_{el} (when $\lambda = 0$) is block diagonal with three identical diagonal blocks (Technical detial of this can be found in A). Therefore only one of these blocks needs to be factorized for a direct solver used.

With the core formulation of Projective Dynamics and its computational structure established, we now turn to the central focus of this dissertation: extending PD to address practical simulation challenges that violate its foundational assumptions. In the next chapter, we present our core simulation framework, designed to maximally exploit the efficiency of PD, along with techniques for incorporating robust collision handling—without sacrificing the key advantage of a constant, prefactored system matrix.

4 OPTIMIZED PROCESSING OF LOCALIZED COLLISIONS IN PROJECTIVE DYNAMICS

This chapter presents a method for efficiently handling contact and collision in volumetric elastic simulations governed by the Projective Dynamics (PD) framework. The primary goal is to enable robust, interactive simulation of high-resolution tetrahedral meshes—containing more than half a million elements—while maintaining computational performance. To achieve this, we focus on models where two key conditions are satisfied (Figure 4.1): first, the subset of the model's surface that is prone to collision must be identifiable in advance; and second, the degrees of freedom (DOFs) associated with this collision-prone region must constitute only a small portion (e.g., approximately 5%) of the total simulation DOFs.

Under these assumptions, we employ a partial Cholesky factorization to effectively decouple the simulation mesh into collision-prone and collision-safe regions. The behavior of the collision-safe region is abstracted into a Schur complement system, enabling localized updates restricted to the collision-prone surface subset. Within this reduced system, we incorporate bilateral penalty-based contact forces and show how these can be updated efficiently on the GPU. Additionally, we pro-

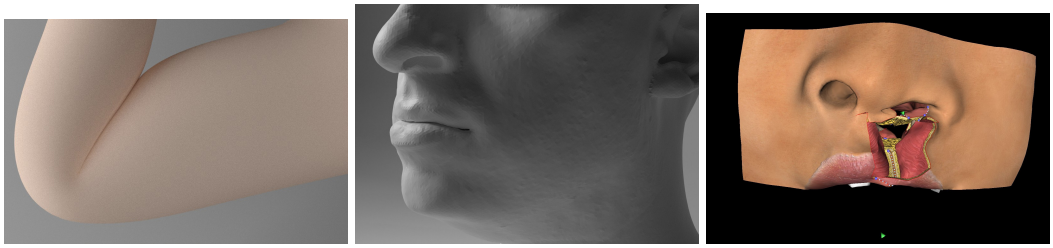

Figure 4.1: Demonstrations of collision handling in our framework. Left: An elbow model, embedded in a tetrahedral simulation mesh with 596K elements. Middle: A face model (644K tetrahedral elements) brought into a self-colliding configuration by articulating the mandible. Right: Simulation of a cleft lip and palate repair in a virtual surgery simulator (468K tetrahedral elements). Persistent collision occurs between the lip and the gum/teeth in the maxilla. Simulation rates for all these examples range between 3-8fps, with full collision handling.

pose a localized update strategy for the rotation variables, analogous to a selective application of the PD local step, which further reduces computational overhead by avoiding unnecessary updates outside the region of interest.

The resulting approach enables stable and high-performance simulation across a range of complex examples, including anatomically detailed models from animation and medical training contexts. By combining system reduction, selective updates, and GPU acceleration, this method significantly extends the scalability of interactive contact simulation within the PD paradigm.

Note on Co-authorship

This chapter is based on joint work with Prof.Eftychios Sifakis, Yutian Tao, Dr. Court Cutting, and Eric Brandt published in *Computer Graphics Forum* [Wang et al. (2021)]. I was responsible for the algorithm design, implementation, and manuscript preparation.

4.1 Introduction

A key advantage of the Projective Dynamics (PD) framework over traditional Newton-based optimization methods lies in its use of an alternating minimization strategy, which offers strong stability guarantees during the simulation of elastic systems. Specifically, PD ensures a monotonic decrease of the system's energy at each iteration, providing a measure of robustness even in scenarios where the solver is terminated before full convergence. This property is particularly advantageous in real-time or interactive settings, where computational resources and timing constraints may prohibit complete convergence within each frame.

Moreover, it has been shown that when PD does converge, it reaches the same minimizer as standard Newton-based approaches. This convergence property makes PD particularly well-suited for interactive applications, where the need for real-time responsiveness often necessitates early termination of iterative solvers. In

such contexts, it is preferable to use a method that maintains stable behavior even when operating under tight performance budgets.

The primary application motivating this work lies in the domain of interactive anatomical simulation, where resolving elastic deformations and handling localized contact events are of central importance. These simulations frequently involve both soft-rigid contact—such as between elastic tissue and fixed skeletal structures—and self-collisions within the deformable body. Our focal use case is a virtual surgical simulation system designed to emulate reconstructive facial procedures, with a particular emphasis on cleft lip and palate repair. This simulator is described in detail in Chapter 5, where we outline its surgical toolset, interactive modeling components, and the simulation challenges that arise during flap-based facial reconstruction. The simulation setting presents unique challenges due to the need for high anatomical accuracy, with resolutions often reaching millimeter scale, and thus serves as a demanding testbed for the collision-handling approach for high resolution simulation mesh developed in this chapter.

Challenges. While Projective Dynamics provides significant advantages in interactive simulation—most notably its robustness under early termination—it is not without important limitations. A central requirement for its strong stability and convergence guarantees is the use of specific material models, such as corotated elasticity [Bouaziz et al. (2014)], or closely related variants [Ichim et al. (2017)]. When employing more general or nonlinear materials, convergence can no longer be taken for granted, and line-search methods or other safeguard techniques—commonly found in Quasi-Newton schemes [Liu et al. (2017)]—may be necessary to ensure reliability.

A more critical challenge arises in the context of collisions. The introduction of contact interactions can directly undermine several of the conditions that make Projective Dynamics efficient and stable. In volumetric quasistatic simulations, collision resolution is most commonly performed through penalty-based methods, wherein zero-restlength springs are introduced to resist interpenetration between surface points [Teschner et al. (2005); McAdams et al. (2011); Mitchell et al. (2015a)].

These springs are typically short-lived and formed dynamically between a surface point detected in collision and its closest opposing contact.

From the perspective of Projective Dynamics, these collision springs behave like additional energy terms that would ideally be folded into the system's global energy minimization. In particular, they would modify the Laplacian-like matrix used in the global step. However, incorporating them directly into the global stiffness matrix poses a significant computational challenge: it invalidates the possibility of using a prefactored direct solver, such as a Cholesky decomposition. This is a serious drawback in interactive scenarios, where simulation meshes with hundreds of thousands of elements can only be run in real time by reusing such prefactorizations.

To preserve the efficiency benefits of direct solvers, several compromises have been proposed. One approach, suggested by the original authors of Projective Dynamics [Bouaziz et al. (2014)], is to construct the global system matrix assuming that *all* potential collision proxies are active, incorporating all internal forces into the system matrix, even if only a subset are currently involved in collision. The right-hand side of the system is then constructed using only those proxies actually engaged. While this approach maintains system stability and avoids expensive matrix updates, system appears significantly damped and slower to react at inactive collision sites and fails to generalize to dynamic self-collisions, where contact patterns evolve unpredictably over time.

An alternative strategy, proposed by Ichim et al. [Ichim et al. (2016)], introduces linear equality constraints to enforce active collision behavior during the global solve. These constraints are incorporated into the minimization using a Schur complement, producing a dense system whose size is determined by the number of active collisions. This method is elegant and flexible, but becomes impractical when the number of active collision proxies grows beyond a few hundred, as the Schur complement must be recomputed at every iteration.

Our proposed method also leverages the Schur complement, but does so in a structurally different way. Rather than introducing it to handle constraints directly, we use it to isolate and localize computation within a pre-designated collision-prone

region. This allows us to retain the performance benefits of prefactorization for the majority of the mesh, while focusing computational updates only where they are most needed. The details of this approach are described in the sections that follow.

4.2 Related Work

Skinning and Collisions

Collision processing for volumetric objects permits a broader range of techniques than those typically used for cloth simulation, owing to the ability of volumetric meshes to recover from temporarily tangled configurations. Among these, implicit geometry representations have been widely adopted for collision detection and response, often paired with penalty-based force models [Teschner et al. (2005); McAdams et al. (2011); Mitchell et al. (2015a)].

Recent developments in interactive skinning techniques have aimed to improve both realism and computational performance. Methods such as implicit skinning [Vaillant et al. (2013)], Delta Mush [Le and Lewis (2019)], and subspace deformation models [Teng et al. (2014)] offer smooth surface deformation and are often integrated with real-time solvers. Several of these approaches draw inspiration from or are combined with the Projective Dynamics (PD) framework [Komaritzan and Botsch (2018, 2019); Lan et al. (2020)], allowing them to inherit its stability and speed benefits.

Position-Based Dynamics (PBD) has also been employed for collision-aware simulations, particularly in the context of skinned or articulated models [Abu Rumman and Fratarcangeli (2015)]. In the domain of muscle-driven animation, volume-preserving primitives such as fiber bundles [Angles et al. (2019)] and simplified anatomy-inspired muscle models [Roussellet et al. (2018)] have been coupled with skinning techniques to model contact under actuation.

More recently, efforts to incorporate frictional contact into PD-inspired solvers have used optimization-based methods such as the Alternating Direction Method of Multipliers (ADMM) [Ly et al. (2020); Daviet (2020)]. These works address general

contact mechanics but often operate under assumptions that are too computationally demanding for high-resolution, interactive surgical simulation.

4.3 Technical Background

Collisions

In the spirit of prior work [Teschner et al. (2005); McAdams et al. (2011); Mitchell et al. (2015a)], we process collisions by sprinkling a number of points on the surface of our volumetric model that we refer to as *collision proxies*. These collision proxies can either be selected among the surface vertices of a conforming volumetric mesh, or simply embedded in the mesh in the sense that each of their locations is barycentrically interpolated from nodes of the containing element. That is, we can represent the location of the j^{th} proxy point, $\vec{p}_j,\ j = 1, \ldots, m$, as a weighted sum of all n mesh vertex locations

$$\vec{p}_j = \sum_{i=1}^{n} w_{(j,i)} \vec{x}_i \quad \text{where} \quad \sum_{i=1}^{n} w_{(j,i)} = 1$$

Note that this vector can be decomposed into a corresponding equation for each component, $\nu = 1, \ldots, 3$:

$$\vec{p}_j^{(\nu)} = \sum_{i=1}^{n} w_{(j,i)} \vec{x}_i^{(\nu)} \quad \text{where} \quad \sum_{i=1}^{n} w_{(j,i)} = 1 \tag{4.1}$$

where $w_{(j,i)}$ is the weight of the i^{th} vertex for the j^{th} proxy. We expect that only 4 components of $w_{(j,i)}$ for any given j are non-zero, corresponding to the vertices of the tetrahedron containing the proxy. At each time step of the simulation, we will make a check to determine if any of these proxies are located in a prohibited region (e.g., inside a kinematic colliding object). Supposing that proxy $\vec{p}_j$ is inside a prohibited region, we will use the geometric representation of the obstacle (typically an implicit surface), to project the proxy location to the colliding region's surface.

We label that point on the obstacle surface $\vec{t}_j$. We then instantiate a short lived, zero-restlength spring connecting $\vec{p}_j$ and $\vec{t}_j$. These springs will contribute to the energy of the system that we seek to minimize. We can write the energy contribution due to collisions concretely as

$$E_{col} = \sum_{j=1}^{m} \frac{c_j}{2} ||\vec{p}_j - \vec{t}_j||_2^2 \cdot \delta_j$$

where c_j is the stiffness coefficient of the j^{th} proxy and δ_j is the indicator function

$$\delta_j = \begin{cases} 0 & j^{th} \text{ proxy is not in collision} \\ 1 & j^{th} \text{ proxy is in collision} \end{cases} \tag{4.2}$$

We can further decompose this by components:

$$E_{col} = \sum_{\nu=1}^{3} \sum_{j=1}^{m} \frac{c_j}{2} \left(\vec{p}_j^{(\nu)} - \vec{t}_j^{(\nu)} \right)^2 \cdot \delta_j \tag{4.3}$$

We can then substitute (4.1) into (4.3) and achieve:

$$E_{col} = \sum_{\nu=1}^{3} \frac{1}{2} \left(W\vec{x}^{(\nu)} - \mathbf{t}^{(\nu)}(\vec{x}) \right)^T C(\vec{x}) \left(W\vec{x}^{(\nu)} - \mathbf{t}^{(\nu)}(\vec{x}) \right) \tag{4.4}$$

where the diagonal matrix $C(\vec{x})$ satisfies $[C(\vec{x})]_{jj} = c_j \cdot \delta_j$ and $W_{ji} = w_{(j,i)}$. Let us highlight two subtle but important points about equation (4.4): First, the only components of the equation that are dependent on $\vec{x}$ are $\mathbf{t}^{(\nu)}(\vec{x})$ and $C(\vec{x})$, with the dependence of the latter being due to proxies being flagged as active or inactive as a function of their placement. Second, the three components (x, y, and z) are separable and independent, just as we saw with the global step of Projective Dynamics. These observations suggest we follow the path of Projective Dynamics

derivation further. We can write E_{col} as an energy-minimization problem:

$$E_{col} = \min_{\vec{t}:\text{collision-free}} \sum_{\nu=1}^{3} \frac{1}{2} \left(W\vec{x}^{(\nu)} - \vec{t}^{(\nu)} \right)^{T} C \left(W\vec{x}^{(\nu)} - \vec{t}^{(\nu)} \right)$$

where the "projections" $\vec{t}_j$ are selected among all *collision-free* locations in the ambient space, as to minimize this energy. Conceptually, this suggests that by freezing $\vec{t}$ and C to specific values (determined by collision detection) as part of the local step, we retain both the stability traits of Projective Dynamics, and the property that this expression becomes a quadratic function. We denote this by $\hat{E}_{col}(\vec{x})$ and this energy term can be folded into the Newton scheme in Equation (3.13) in the global step.

4.4 Proposed Method

We present our method by first partitioning our mesh into a collision-prone and a collision-safe region. We then use a Schur complement method to craft a numerical solution that concentrates on the collision prone region. Finally, we present a nested iteration that can refine the solution in the vicinity of collisions, at low cost.

Collision in Projective Dynamics

The power of Projective Dynamics is largely due to the ability to pre-factorize the system stiffness matrix using a Cholesky Factorization. Once this matrix is factorized, performing the global step of Projective Dynamics only incurs the cost of a single forward and backward substitution. However, as we discussed, when the simulation involves collisions, the system matrix changes at each step, compromising the ability to use a constant, pre-factored matrix. Let us start by taking a closer look at the total energy equation of the global step, which is the sum of the energy due to elastic deformation and the energy due to collisions:

$$\hat{E}_{tot} = \hat{E}_{el} + \hat{E}_{col} \tag{4.5}$$

Differentiating (4.5) once, we see the total forces (the complete right hand side of (3.14) :

$$-\frac{\partial E_{tot}}{\partial \vec{x}} = \vec{\mathbf{f}}_{tot} = \vec{\mathbf{f}}_{el}(\vec{\mathbf{x}}) - W^T C(\vec{\mathbf{x}})(W\vec{\mathbf{x}} + \vec{\mathbf{t}})$$

and differentiating again we see the complete left hand side of (3.14):

$$\frac{\partial^2 E_{tot}}{\partial \vec{x}^2} = K + W^T C(\vec{\mathbf{x}})W$$

Unfortunately, it is the case that our constant matrix used in the global step has been polluted by terms that depend on $\vec{\mathbf{x}}$. Given that the matrices we are targeting will have in the order of 10^5 nodal degrees of freedom, we cannot tolerate the re-factorization cost at each time step for an interactive application. A second option is to use an iterative approach to solve the system without factorization. In section 4.5 we discuss when this is appropriate, but also note that convergence may then suffer for high resolution models. Perhaps another option would be to observe that the matrix $W^T C(\vec{\mathbf{x}})W$ that depends on $\vec{\mathbf{x}}$ is low rank. The problem with this is that even our "low rank" matrix has a rank in the hundreds to low-thousands for typical simulations. This would quickly yield an untenable proposition trying to manage such an effort with a low-rank update algorithm, especially with how frequently the matrix changes.

Domain Partitioning for the Global Step

Motivated by these observations, we craft an approach that leverages our modeling hypotheses, namely:

1. The fraction of the mesh prone to collision is a small subset ($< 5\%$) of the simulated model, and

2. The region where collisions may occur can be known a-priori.

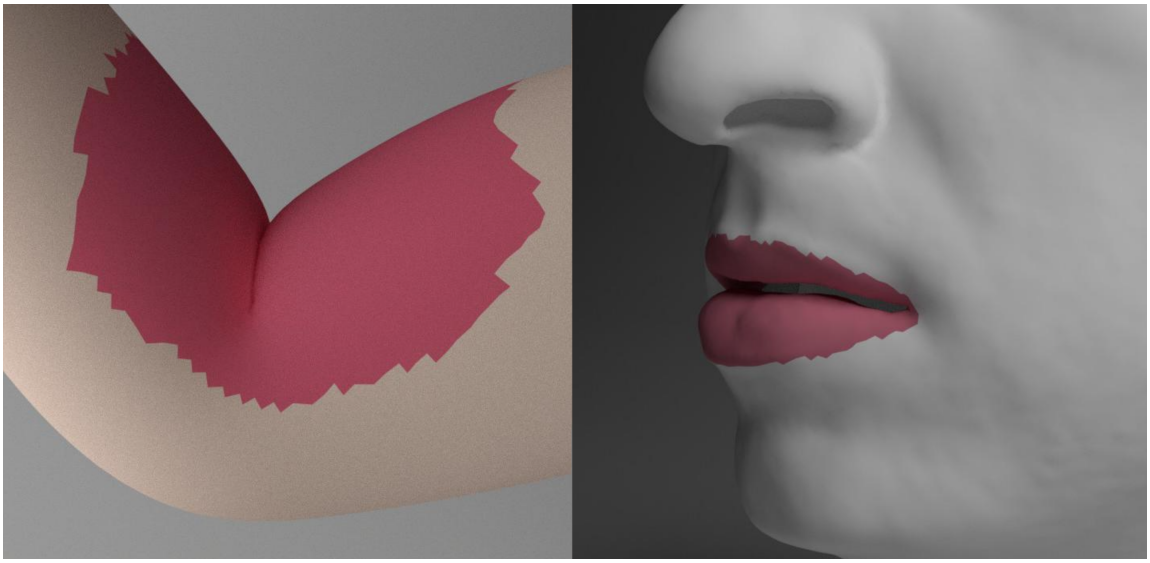

Figure 4.2: Our method requires an a-priori designation of a fraction of nodes as "collision-prone", highlighted here in red. For efficiency we limit such nodes to a small subset (e.g. 5%) of the mesh.

Consider the rank of the components of the global step matrix:

$$\overbrace{K + \underbrace{W^T C W}_{\text{rank } m}}^{\text{rank } n}$$

Here, n is the number of simulation mesh vertices. The part of the matrix that is changing, however, is a much smaller $m \times m$ submatrix. An upper bound on m will be the cardinality of the union of vertices of tetrahedral elements that contain a collision proxy. As a practical matter, in our simulation of the face, m is the number of degrees of freedom of the tetrahedral elements surrounding the lips, where in the arm model the same area is localized around the inner fold of the elbow joint, as shown in Figure 4.2. In the experimental examples presented in section 4.5, models typical have in the order of 500K tetrahedra, 100K vertices, of which 1000-3000 vertices fit the criteria of anchoring a tetrahedral element that contains a collision proxy.

Referring to Figure 4.3 as an example, we can partition the full set of vertices, $\vec{x}$, into two subsets: $\vec{x} = \begin{pmatrix} \vec{x}_1^T & \vec{x}_2^T \end{pmatrix}^T$ where $\vec{x}_1$ contains the nodes *not* in the immediate vicinity of collision proxies, and $\vec{x}_2$ contains the nodes that *are* in the immediate vicinity of collisions. To further clarify, in Figure 4.3, n is the total number of all vertices ($\vec{x}_1 \cup \vec{x}_2$), and m is the number of vertices in $\vec{x}_2$. Then, we re-write the global step equation as follows $(K + W^T C W)\delta\vec{x} = \vec{f}_{tot}$ in block form:

$$\begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} + C_{22}(\vec{x}) \end{pmatrix} \begin{pmatrix} \delta\vec{x}_1 \\ \delta\vec{x}_2 \end{pmatrix} = \begin{pmatrix} \vec{f}_1 \\ \vec{f}_2 + d_2(\vec{x}) \end{pmatrix} \tag{4.6}$$

where $C_{22}(\vec{x}) = W^T C(\vec{x}) W$ and d_2 are the force components induced by collisions. Based on the relative sizes of $\vec{x}_1$ and $\vec{x}_2$, we point out that the entire matrix is largely unchanged and remains constant, with only a very small subset of the matrix being dependent on the current vertex locations. Next, we capitalize on this structure and relative sizes by using a Schur complement method.

Partial Cholesky Schur Complement Factorization

Consider a linear system, $Ax = b$, where A is symmetric. Partition A, x, and b such that the system can be written in block format:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{4.7}$$

Suppose that A_{11} has the Cholesky factorization $A_{11} = L_1 L_1^T$. Careful multiplication will verify the following factorization of A:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_1 & 0 \\ A_{21} L_1^{-T} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \Sigma \end{pmatrix} \begin{pmatrix} L_1^T & L_1^{-1} A_{12} \\ 0 & I \end{pmatrix} \tag{4.8}$$

where $\Sigma = A_{22} - A_{21} A_{11}^{-1} A_{12}$ is known as the Schur complement. It is important to realize that the factorization in equation (4.8) is nothing more than a partial Cholesky factorization, and is typically an intermediate step in the full Cholesky

factorization of. The Intel® MKL PARDISO library, which we use, can be invoked as to compute exactly such a factorization, providing the user with the express value of the Schur complement while retaining the partial triangular factors in its internal representation.

We should note that our partitioning of nodes into the subsets $\vec{x}_1$ and $\vec{x}_2$ does incur some slight sub-optimality relative to the stock (non-Schur) Cholesky factorization, by constraining the degrees of freedom in $\vec{x}_1$ to appear strictly before those in $\vec{x}_2$. Our experiments however indicate that any deterioration in sparsity of the resulting factors was less than 10% in all of our examples.

Performance Optimization

The Intel® MKL PARDISO sparse factorization library provides a highly optimized CPU-based implementation of the partial factorization shown in equation (4.8) and discussed above. The user provides as input the symmetric system matrix and the degrees of freedom that are to be maintained after the factorization. The library provides as output the (dense) Schur complement Σ matrix, and maintains the partial triangular factors in internal representation.

Now our system shown in (4.7), after factorization, becomes:

$$\underbrace{\begin{pmatrix} L_1 & 0 \\ A_{21}L_1^{-\mathsf{T}} & I \end{pmatrix}}_{\text{lower-tri}} \begin{pmatrix} I & 0 \\ 0 & \Sigma \end{pmatrix} \underbrace{\begin{pmatrix} L_1^{\mathsf{T}} & L_1^{-1}A_{12} \\ 0 & I \end{pmatrix}}_{\text{upper-tri}} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix} \tag{4.9}$$

We can solve this in a series of steps:

Step 1: Solve the lower triangular system:

$$\begin{pmatrix} L_1 & 0 \\ A_{21}L_1^{-\mathsf{T}} & I \end{pmatrix} \begin{pmatrix} \vec{y}_1 \\ \vec{y}_2 \end{pmatrix} = \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix} \tag{4.10}$$

Being a lower triangular matrix, this can be solved quickly by forward substitution on the CPU using Intel® MKL PARDISO optimized forward substitution algorithm (PARDISO "phase 331").

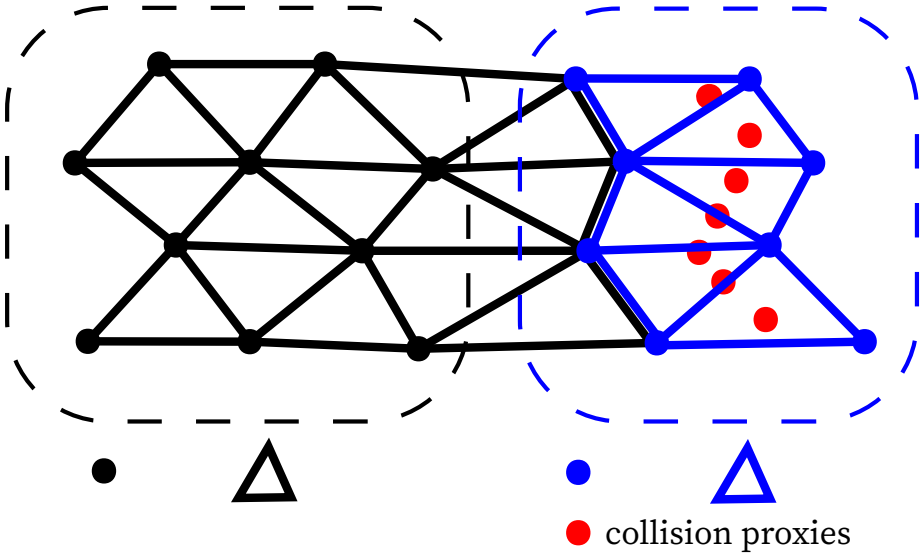

Figure 4.3: Simulation mesh partitioning. Collision proxies shown in red. Elements that embed collision proxies form the set $\mathcal{E}_\beta$ (blue), while their complement is the collision-safe region $\mathcal{E}_\alpha$ (black). The collision-prone nodes $\vec{x}_2$ (blue) are those that appear in elements of $\mathcal{E}_\beta$, while all other (collision-safe) nodes are grouped in $\vec{x}_1$ (black).

Step 2: Solve the system:

$$\begin{pmatrix} I & 0 \\ 0 & \Sigma \end{pmatrix} \begin{pmatrix} \vec{z}_1 \\ \vec{z}_2 \end{pmatrix} = \begin{pmatrix} \vec{y}_1 \\ \vec{y}_2 \end{pmatrix} \tag{4.11}$$

It is trivial to see that $\vec{z}_1 = \vec{y}_1$ and that $\vec{z}_2 = \Sigma^{-1}\vec{y}_2$. We will momentarily defer discussion of how to solve this until completing the description of the full solution.

Step 3: Finally, Solve the system:

$$\begin{pmatrix} L_1^T & L_1^{-1}A_{12} \\ 0 & I \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \begin{pmatrix} \vec{z}_1 \\ \vec{z}_2 \end{pmatrix} \tag{4.12}$$

Being an upper triangular matrix, this can be solved quickly by backward substitution on the CPU using MKL PARDISO optimized backward substitution algorithm

("phase 333").

At first glance, a concern might be that since the overall matrix in our application is changing with each time step due to collisions, that we may still have to recompute this partial factorization at each timestep to produce a new Σ. Fortunately this is not the case (Figure 4.4). To see this, refer to (4.6) and consider the Schur complement of the K matrix *without* any collision forces. *Without* collision forces, the Schur complement of the block K matrix would be

$$\Sigma_{\text{no-col}} = K_{22} - K_{21}K_{11}^{-1}K_{12} \tag{4.13}$$

In the presence of collisions, the term K_{22} has been replaced by $K_{22}+C_{22}(\vec{x})$. Because K_{22} only appears on the right hand side of (4.13) as a lone term, we can see that in the presence of collisions, we can simply add the $C_{22}(\vec{x})$ term to yield:

$$\begin{aligned}\Sigma_{\text{col}} &= K_{22} - K_{21}K_{11}^{-1}K_{12} + C_{22}(\vec{x}) \\ &= \Sigma_{\text{no-col}} + C_{22}(\vec{x})\end{aligned}$$

This means that we can compute the partial factorization only once in the absence of collisions and retain a $\Sigma_{\text{no-col}}$ matrix, to which we can add $C_{22}(\vec{x})$ at each time step. This addition is performed as a purely *additive* update to the Schur Complement, and is very efficient. Now knowing that it is easy to produce the correct Σ matrix at each timestep, we return to describing an efficient solution to $\Sigma\vec{z}_2 = \vec{y}_2$. Recalling that the expected size of Σ is approximately 1000-3000 degrees of freedom, we are presented with a slightly surprising opportunity that one might otherwise overlook. Although for the *global, sparse matrix* it is not practical to repeat a factorization every time its entries change, for the *local, dense* matrix Σ the refactorization is a perfectly realistic and efficient option.

To support this point, let us explore the cost of factorizing Σ and directly solving $\Sigma\vec{z}_2 = \vec{y}_2$. For purposes of illustration, we will consider a typical Σ in our simulation having $m \approx 2000$ degrees of freedom. (dimension $\approx 2000 \times 2000$). A full Cholesky factorization requires $\frac{1}{6}m^3$ floating point operations (FLOPS), or in our case will

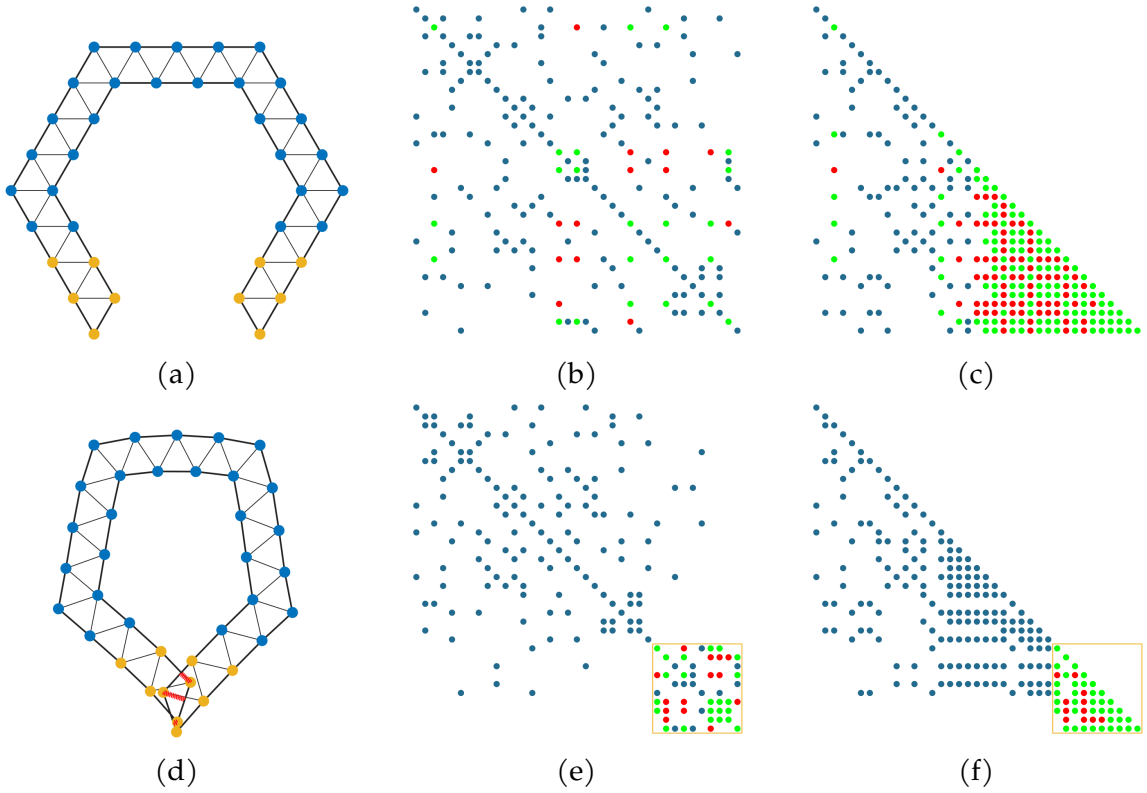

Figure 4.4: (a) Rest configuration of the mesh. The nodes that will not be inflicted by collisions are colored blue; nodes that could potentially involve in collision are colored yellow. (b) Change of sparsity pattern of stiffness matrix K under original ordering. New entries that are created due to the addition of the collision springs are colored red; Entries that are in the original stiffness matrix but has changed numerical value due to collision springs are colored green. (c) Sparsity pattern of the cholesky factor L n under original ordering. (d) Deformed configuration of the mesh, with same node coloring scheme as in (a). Zero-rest-length springs (red) are added to the interpenetrating nodes to push them to the surface of the mesh. (e) Change of sparsity pattern of stiffness matrix K under with our reordering. Notice that all the changes in the sparsity patterns of the stiffness matrix are now confined to the lower right corner of the matrix. (f) Change of sparsity pattern of cholesky factor L under with our reordering. Because changes in the sparsity patterns of the stiffness matrix are now confined to the lower right corner, the changes in the cholesky factor are also confined.

be $\approx \frac{8}{3}$ billion floating point operations (GFLOPS). Modern workstation-grade CPUs are capable of 2+ trillion floating point operations (TFLOPS) per second, and modern high end GPUs are capable of approximately 12 TFLOPS. This suggests we have the ability to factorize such a system in a budget of low number of milliseconds; such opportunity is not afforded to *sparse* matrices, as their processing is often bound by memory bandwidth. For such sizes of dense matrices however, the cubic computational complexity is well counterbalanced by the (typical 100:1) ratio of possible arithmetic computations per memory access on CPUs or GPUs.

With this "order of magnitude" calculus suggesting that we have a promising approach to achieving the desired performance, we describe our implementation of Step 2. We solve this system on the GPU. At the start of simulation, we move the $\Sigma_{\text{no-col}}$ matrix to the GPU memory. At each timestep, we move the $C(\vec{\boldsymbol{x}})$ matrix and the $\vec{\mathbf{y}}_2$ vector to the GPU. In case of self-collisions, we may need to also transmit to the GPU the embedding weight matrix W. Recall that $C(\vec{\boldsymbol{x}})$ is a diagonal matrix, so the bandwidth per timestep is small, and similarly W is sparse. We then use highly efficient stock algorithms available in NVIDIA cuSPARSE and cuSOLVER libraries to a) perform the rank-k update on the pre-calculated Schur complement matrix (using cusparseScsrgemm2), b) refactorize it on the fly (with cusolverDnSpotrf), and c) perform the forward and backward substitution to provide the solution (cusolverDnSpotrs), $\vec{\mathbf{z}}_2$, which we stream back to main memory and proceed with Step 3 to complete the solution steps for the current time step.

Further Optimization of the Solution

A frequent observation in simulation of volumetric objects with collisions is that the non-linear, highly volatile penalty terms are the main contributor to the need for a large number of iterations at each time step to achieve convergence. In particular, it is the changing nature of collision proxies alternating between being active and inactive during iteration. Although all of these volatile behaviors are localized, traditionally the cost that we pay is global.

In this section, we take the opportunity to investigate the possibility to com-

ALGORITHM 1: Optimized solve with collisions

preliminary: Schur-factorize:

$$\begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22}^{(\alpha)} \end{pmatrix} = \begin{pmatrix} L_1 & 0 \\ K_{21}L_1^{-\mathsf{T}} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \Sigma^{(\alpha)} \end{pmatrix} \begin{pmatrix} L_1^{\mathsf{T}} & L_1^{-1}K_{12} \\ 0 & I \end{pmatrix}$$

input : partially factorized stiffness matrix with Schur Complement; updated Dirichlet node positions

output: Correction amount $\vec{\mathbf{u}} = \begin{pmatrix} \vec{\mathbf{u}}_1^{\mathsf{T}} & \vec{\mathbf{u}}_2^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}$

for $i \leftarrow 1$ **to** outer_iters **do**

1. $\mathbf{R}_\alpha \leftarrow$ `LocalStep ()`
2. $\begin{pmatrix} \vec{\mathbf{f}}_{1;\alpha}^{\mathsf{T}} & \vec{\mathbf{f}}_{2;\alpha}^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}} \leftarrow$ `ComputeForces () // only` $\hat{E}_\alpha$
3. Solve $\begin{pmatrix} L_1 & 0 \\ K_{21}L_1^{-\mathsf{T}} & I \end{pmatrix} \begin{pmatrix} \vec{\mathbf{y}}_1 \\ \vec{\mathbf{y}}_2 \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{f}}_{1;\alpha} \\ \vec{\mathbf{f}}_{2;\alpha} \end{pmatrix}$ via ForwardSub

`// We expect:` $\begin{pmatrix} \vec{\mathbf{y}}_1 \\ \vec{\mathbf{y}}_2 \end{pmatrix} = \begin{pmatrix} L_1^{-1}\vec{\mathbf{f}}_{1;\alpha} \\ \vec{\mathbf{f}}_{2;\alpha} - A_{21}A_{11}^{-1}\vec{\mathbf{f}}_{1;\alpha} \end{pmatrix}$

3.1 $\tilde{\mathbf{f}}_{2;\alpha} = \vec{\mathbf{y}}_2$

4. **for** $j \leftarrow 1$ **to** inner_iters **do**
   - 4.1 $C(\vec{\mathbf{x}_2}) \leftarrow$ `DetectCollisions` $(\vec{\mathbf{x}}_2)$
   - 4.2. $\mathbf{R}_\beta \leftarrow$ `LocalStep ()`
   - 2. $\vec{\mathbf{f}}_{2;\beta} \leftarrow$ `ComputeForces () // only` $\hat{E}_\beta$
   - 4.3 $H \leftarrow \Sigma^{(\alpha)} + K_{22}^{(\beta)} + C$
   - 4.4 $\vec{\mathbf{g}} \leftarrow \tilde{\mathbf{f}}_{2;\alpha} + \vec{\mathbf{f}}_{2;\beta} + \vec{\mathbf{f}}_{\text{col}}(\vec{\mathbf{x}}_2)$
   - 4.5 Solve $H\vec{\mathbf{u}}_2 = \vec{\mathbf{g}}$
   - 4.6 $\vec{\mathbf{x}}_2 += \vec{\mathbf{u}}_2$
   - 4.7 $\tilde{\mathbf{f}}_{2;\alpha} -= \Sigma^{(\alpha)}\vec{\mathbf{u}}_2$

   end
5. Solve $L_1^{\mathsf{T}}\vec{\mathbf{u}}_1 = \vec{\mathbf{y}}_1 - L_1^{\mathsf{T}}K_{12}\vec{\mathbf{u}}_2$ using BackwardSub

end

pletely restrict the iterative computation so that it only takes place in the immediate vicinity of the collision prone region. To begin, refer again to Figure 4.3, observing the two partitions:

1. The elements are partitioned into black elements, $\mathcal{E}_\alpha$, which are elements that do not contain collision proxies, and blue elements, $\mathcal{E}_\beta$, which are elements

that *do* contain collision proxies.

2. Vertices of the collision-prone elements $\mathcal{E}_\alpha$ will be labeled the collision-prone set $\vec{\mathbf{x}}_2$ (in blue); their complement will be the collision-safe nodes $\vec{\mathbf{x}}_1$ (in black).

In this partitioning the element set $\mathcal{E}_\alpha$ is associated with vertices from *both* $\vec{\mathbf{x}}_1$ and $\vec{\mathbf{x}}_2$, while $\mathcal{E}_\beta$ is associated with vertices *only* from $\vec{\mathbf{x}}_2$. We also refer back to equation (4.5) that defines the total energy, recalling that the quasistatic solution can be written as a minimization problem under the context of Projective Dynamics (as in (3.12)):

$$\min_{\vec{\mathbf{x}}} \mathsf{E}(\vec{\mathbf{x}}) = \min_{\vec{\mathbf{x}}} \left(\mathsf{E}_{\text{el}}(\vec{\mathbf{x}}) + \mathsf{E}_{\text{col}}(\vec{\mathbf{x}})\right) \tag{4.14}$$

$$= \min_{\vec{\mathbf{x}},\mathbf{R}} \left(\hat{\mathsf{E}}_{\text{el}}(\vec{\mathbf{x}},\mathbf{R}) + \mathsf{E}_{\text{col}}(\vec{\mathbf{x_2}})\right) \tag{4.15}$$

$$= \min_{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\mathbf{R}_\alpha,\mathbf{R}_\beta} \left(\hat{\mathsf{E}}_\alpha(\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\mathbf{R}_\alpha) + \hat{\mathsf{E}}_\beta(\vec{\mathbf{x}}_2,\mathbf{R}_\beta) + \mathsf{E}_{\text{col}}(\vec{\mathbf{x}}_2)\right) \tag{4.16}$$

The final line above separates the equation into contributions from $\mathcal{E}_\alpha$ and $\mathcal{E}_\beta$, being careful to note that the first (α) term is a function of $\vec{\mathbf{x}}_1$ and $\vec{\mathbf{x}}_2$ while the second (β) term is a function only of $\vec{\mathbf{x_2}}$.

To see how we can solve the global step in a nested iteration, first assume that the $\mathbf{R}_\alpha$ and $\mathbf{R}_\beta$ have been fixed by the local step. We can then rewrite (4.16) with fixed rotations as a nested minimization:

$$\mathsf{E}(\vec{\mathbf{x}}) = \min_{\vec{\mathbf{x}}_2} \Big\{ \underbrace{\min_{\vec{\mathbf{x}}_1} \hat{\mathsf{E}}_\alpha(\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\mathbf{R}_\alpha)}_{\tilde{\mathsf{E}}_\alpha(\vec{\mathbf{x}}_2,\mathbf{R}_\alpha)} + \hat{\mathsf{E}}_\beta(\vec{\mathbf{x}}_2,\mathbf{R}_\beta) + \mathsf{E}_{\text{col}}(\vec{\mathbf{x}}_2) \Big\} \tag{4.17}$$

Our method solves (4.16) by a nested iteration that leverages the opportunity to do additional processing on the collision affected region (β) in an inner loop without compromising the the correctness of the solution in the non-collision affected region (α). Our approach involves these steps:

Preamble of outer loop We optimize only rotations $\mathbf{R}_\alpha$ in the collision-safe region, keeping all other variables fixed.

Inner loop Treating $\mathbf{R}_\alpha$ as fixed, we use the Schur Complement to express the minimum (over $\vec{\mathbf{x}}_1$) of $\hat{\mathrm{E}}_\alpha$ as a function, $\tilde{\mathrm{E}}_\alpha$, of only $\vec{\mathbf{x}}_2$ and $\mathbf{R}_\alpha$ (the latter being a constant). This is equivalent to the elimination of $\mathbf{x}_1$ from the global step via the Schur Complement, as described in the previous section. The resulting energy is only a function of $\vec{\mathbf{x}}_2$ and $\mathbf{R}_\beta$ at this point, and we iterate on it in the style of Projective Dynamics – freezing each of $\vec{\mathbf{x}}_2$ and $\mathbf{R}_\beta$ and optimizing over the other – combined with collision detection and update of proxies at the local step.

Conclusion of outer loop We reconstruct the solution for the collision-safe region via backward substitution.

This nested iterative solution procedure is captured in Algorithm 1, where the specific utilization of the MKL PARDISO library is highlighted. From the inner loop, update of the Schur-derived Hessian matrix H, its dense factorization, and the solution for the local update $\vec{\mathbf{u}}_2$, which are are hosted on the GPU. As noted above, we perform steps 4.2 through 4.4 of the algorithm on the GPU. We point out that we make the choice of skipping the calculation of $\mathbf{R}_\beta$ in step 1 and instead doing it inside the inner loop before step 4.1. Updating $\mathbf{R}_\beta$ as part of each inner iteration will improve the rate of convergence, with the minimal extra cost of updating the small number of collision-affected rotations during each inner iteration. In our experience, this extra computation pays off in convergence rates as Figure 4.6 illustrates.

4.5 Results and Evaluation

We demonstrate our method on three quasistatic simulation scenarios, all of which achieve interactive performance with resolutions in the order of half million elements. On two of the test we perform resolution studies demonstrating the necessities of the high resolution volumetric mesh we used. We also perform a comparison of our direct solver to a GPU optimized iterative scheme [Komaritzan and Botsch (2019)] and an alternate direct, factorization-based method [Komaritzan and Botsch (2018)]. All our tests were on an Intel Core i9-9940X CPU @ 3.30GHz and a NVIDIA

		Elbow Flex	Face Simulation	Virtual Surgery
Outer Loop - Preamble				
CPU	PD Local Step (collision-safe region) (includes Rotation Update and and computing RHS of PD system)	4.69ms	6.14ms	5.38ms
CPU	Forward Substitution (MKL PARDISO stage 331)	63.73ms	55.29ms	37.89ms
Inner Loop (collision-prone region)				
CPU	Collision Detection	32.81ms	9.07ms	0.51ms
CPU	Update of Contact Constraints (includes update of weight matrix W and dispatch to GPU)	0.65ms	0.26ms	0.12ms
GPU	Rank-k Update of Schur matrix	5.40ms	5.04ms	1.05ms
GPU	Dense Cholesky of Schur matrix	8.20ms	3.02ms	2.00ms
CPU	Update Right-Hand-Side of Schur system (includes Rotation Update on collision-prone region)	3.48ms	2.16ms	1.33ms
GPU	Solve Schur System (via substitution)	3.22ms	1.80ms	1.10ms
Outer Loop - Conclusion				
CPU	Backward Substitution (MKL PARDISO stage 333)	69.15ms	58.70ms	39.61ms
	PD Iteration Total (1 inner loop)	191.33ms	141.45ms	88.99ms
	PD Iteration Total (5 inner loops)	406.37ms	226.71ms	113.43ms
	Total Tetrahedra in Mesh	596,064	644,486	467,642
	Total Simulation Nodes	111,666	123,784	92,278
	Nodes in Collision-Prone Region	3,933	1,846	1,376

Figure 4.5: Timing results of our three featured benchmarks.

TITAN X GPU. See Figure 4.5 for detailed benchmark results. In our implementation, all outer loop calculations are run on the CPU including the Projective Dynamics local step, forward substitution and backward substitution. Inside the inner loop, we perform collision detection, update contact constraints, and update the right-hand side of the Schur system on the CPU. The other steps, including rank-k update and dense Cholesky factorization are done on the GPU. We leverage AVX512 vectorization for the local step on the CPU, including update of rotations and force computation.

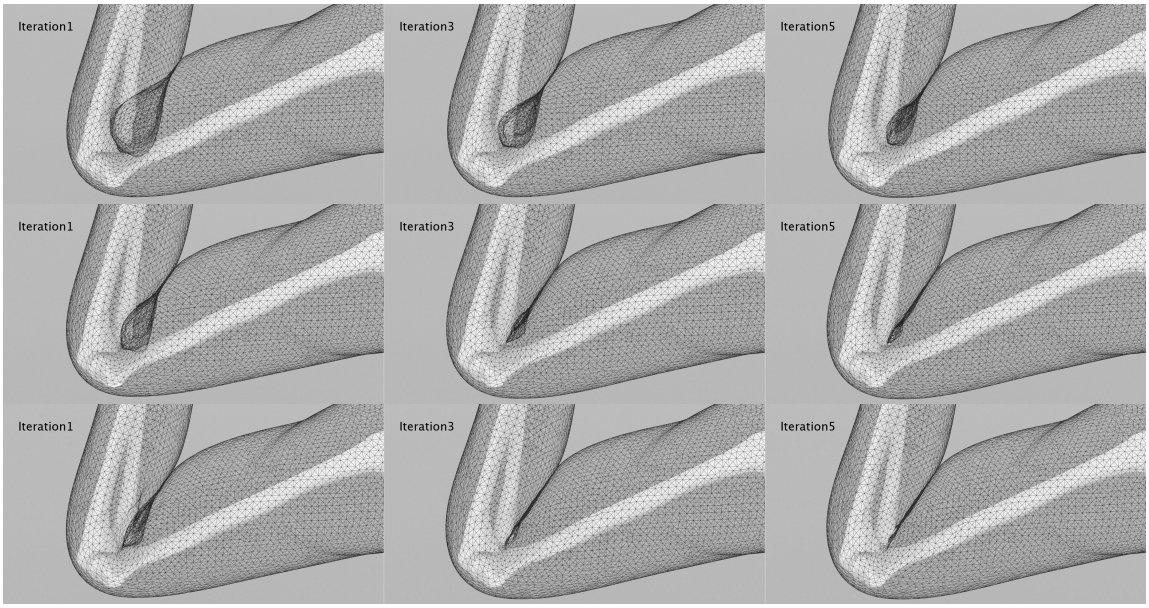

Figure 4.6: The arm is bent with collisions disabled, and then collisions are turned on, forcing the mesh to untangle. One inner iteration per PD loop is illustrated on the left, 5 iterations on the right.

Arm Flexing at Elbow Joint

The first example we examine involves simulating a human arm bending at the elbow joint. In this case, self collisions only occur on the flesh surface on the interior side of the joint. The arm model is embedded in a tetrahedron mesh with 644,486 elements and 111,666 simulation nodes. The potential colliding region is predetermined and marked with collision proxies, which results in 3,933 collision-prone nodes - about 3.5% of the total simulation nodes, as seen in Figure 4.2 (left). Skeletal bones are attached to the simulation mesh by zero restlength springs uniformly sampled over the bone surface. Collision detection and response uses rest-pose levelset representations [McAdams et al. (2011)] of the arm to detect interpenetration and instance collision springs. Using a direct solver for the global step allows our solver to produce a smooth and visually converged animation even with a single Projective Dynamics loop per animation frame. We experimented with both 1 inner-loop of localized update of element rotations in the collision-prone

region per global PD iteration, and 3 or 5 inner-loops which only modestly adds to the solver cost (most of the added cost comes from the repeated detection step) but significantly improves the convergence in strenuous contact cases, as the test in Figure 4.6 where the elbow is brought to a sharp angle with collisions disabled, and attempts to disentangle from this state when collision response is again enabled. Even in challenging frames of this animation, we achieve 5fps with 1 inner loop, and 2.5fps with 5 inner loops.

Face Simulation with Self Collision at Lips

Next, as seen in Figure 4.7, we demonstrate a face simulation, where self collisions occur purely around the lip region. With 644,486 embedding tetrahedral elements, there are 123,784 simulation nodes, only 1,846 of which are contained in the collision prone region. Here the ratio of collision-prone nodes is 1.5%. We achieve 5-7fps. We have used this example as an opportunity to elucidate the value of using as high of a resolution (644K elements) as our algorithm interactively supports. First, we simulated the same jaw grinding motion on a coarser volumetric mesh with 152,316 tetrahera, and 31,834 mesh vertices. We observe that the lower embedding resolution creates occasional contact artifacts on the sides of the mouth region, where the lower available resolution does not allow the corners of the lips to come into a tight closure. We also demonstrate an instance where the high mesh resolution is used to create a high-detail facial expression using "shape targeting" constraints [Klár et al. (2020); Ichim et al. (2017)] which are fully compatible and integrated in our Projective Dynamics and collision handling pipeline.

Surgical Cleft Lip and Palate Simulation

Finally, we apply our algorithm in a virtual surgery simulator where a volumetric facial flesh mesh from a patient with a cleft lip and palate defect is discretized into 503,910 tetrahedra and 97,249 simulation nodes. At some point during surgical manipulation, tissue is excised, leaving behind a simulation mesh with 467K tetrahedra and 92K vertices, as reported in Figure 4.5. In our test, only collisions

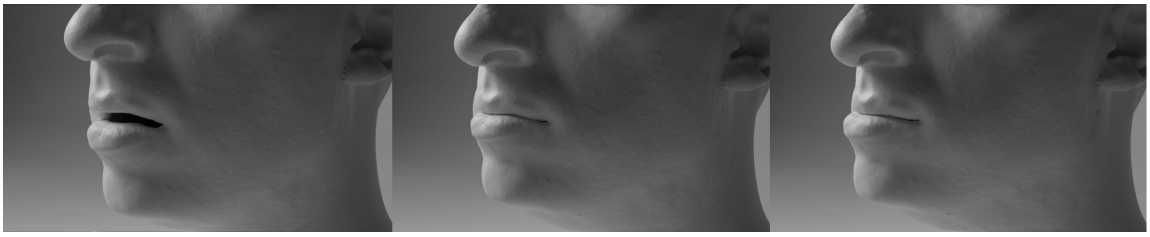

Figure 4.7: As the jaw moves, the lips engage in self-collision which is efficiently resolved. This face model contains 644k tetrahedral elements and 124k nodes, of which 1,846 are collision-prone.

between the deformable flesh and the rigid teeth and maxilla are processed, as the surgical repair being modeled relies on comprehensive suturing to create the final repair. With this hypothesis, we mark 1,434 collision prone nodes in the designated area inside of the lip. Screen shots are shown in Figure 4.8.

We note that in the context of an interactive virtual surgery simulator, where the topology of the model is changing in the course of manipulation and the placement of incisions is not known ahead of time, it is particularly important to avail the model of adequate embedding resolution to allow incisions to open realistically. We tested an instance where the same sequence of manipulations, including incisions, has been repeated on both our full-resolution model, and a coarser one with 49K tetrahedra. Qualitative differences are visible, including incisions that are precluded from opening as widely as in the full-resolution models, and reduced detail in the simulation of retracted tissue.

For this simulation it was essential to capture the biphasic response of skin, which becomes notably more stiff once a strain limit has been reached. We model this effect by adapting the energy density function Ψ to incorporate a strain-limiting term, as follows:

$$\Psi'(F) = \min_{Q \in S} \mu' \|F - Q\|_F^2 + \min_{R \in SO(3)} \mu \|F - R\|_F^2$$

$$S = \{A \in \mathbb{R}^{3\times 3} \text{ s.t } \sigma_{min} < \sigma_i(A) < \sigma_{max}\}$$

Where σ_{min}, σ_{max} are the lower and upper limits that we wish to allow our principal strain to assume. Scalar μ' is the increased stiffness when tissue enters the biphasic

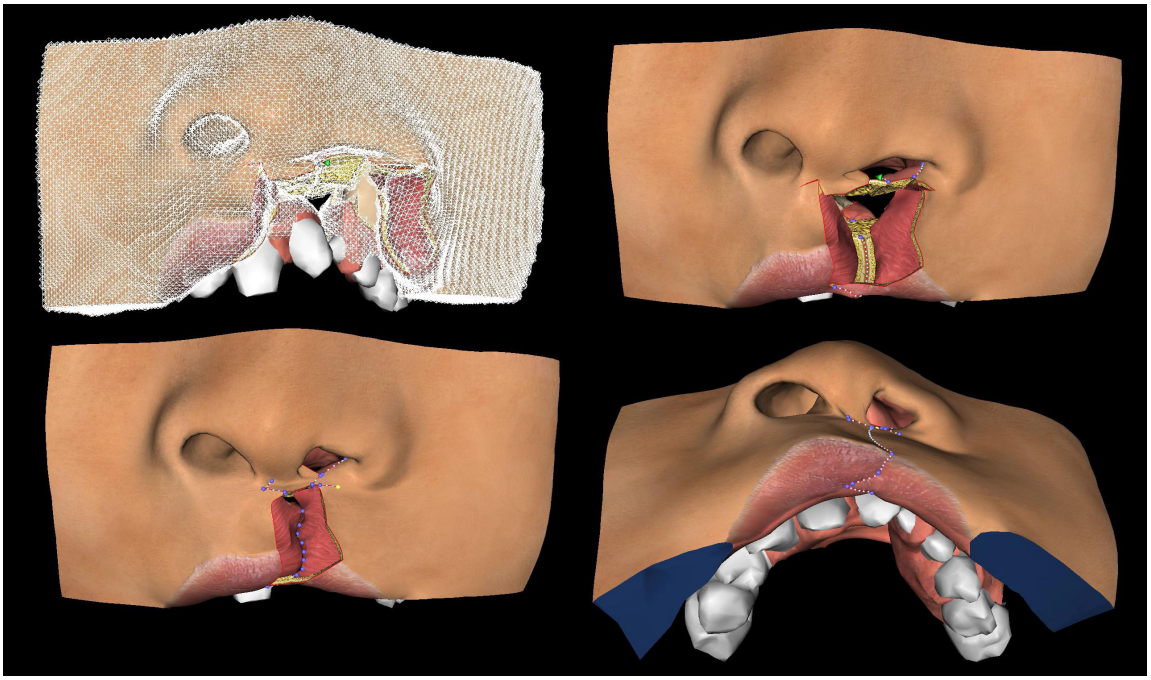

Figure 4.8: Screenshots of an interactive cleft lip surgery simulator.

regime. Minimization of the resulting energy proceeds in the same fashion as the standard Projective Dyanmics paradigm; in the local step, the matrix Q is projected to the allowable set by computing the Singular Value Decomposition and clamping the singular values to the interval $[\sigma_{min}, \sigma_{max}]$. In the global step, the matrices R and Q per element are frozen, and the resulting quadratic energy is minimized.

Comparison with Projective Skinning [Komaritzan and Botsch (2018)]

We performed a comparison with the Projective Skinning technique proposed by [Komaritzan and Botsch (2018)]. Their method also seeks to incorporate collision processing in a Projective Dynamics solver that uses direct, factorization-based solvers for the global step. Similar to assumptions that we make, they localize the possible collision events to known regions, typically around joints of an articulated skeleton.In their work, two variants of the global step matrix are precomuted and

factorized; one corresponding to no collisions being activated, and a different one with every possible collision proxy presumed active. Then, for each animation frame, a number of solve steps can be iterated without collision detection, using the collision-free matrix, and then collision detection can be turned on for a number of additional iterations, using the matrix with all collision proxies activated. For collision constraints, unilateral constraints are used to avoid the changes in the sparsity pattern of the stiffness matrix due to coupling in the bilateral constraint case, where the nodes involved will be in constant change due to sliding during collision resolution. Then, heuristics are used in local steps to project the interpenetrating proxies to non-colliding positions and non interpenetrating ones to their positions in the previous PD steps. Their work utilizes a conforming tetrahedral mesh specially generated for the purpose of skinning and therefore all collision proxies are identified with simulation nodes. In our test, we extended their collision constraints to include support for embedded collision proxies. For projection targets of the interpenetrating collision proxies, we used the midpoint of the proxy and its projection on the offending surface, using a reference-pose levelset representation to compute this surface projection [McAdams et al. (2011)].

A plausible convergence can be obtained during the iterations that incorporate collision handling, even if the global step matrix preemptively includes all candidate collisions. However, for the non-colliding proxies, the projection towards positions in previous PD iterations will create drag in the elastic response of the non-colliding region. With a more careful examination of the final configuration of the result, it can be seen that the result from collision handling scheme in Projective Skinning deviates from the fully converged shape that our method computes; this is due to the fact that the inclusion of PD iterations with collisions disabled at the beginning of each animation frame can move the solution away from a converged state, even if no skeletal motion has occurred. Finally, our method is no more expensive from a computational standpoint (for the same number of iterations) as the bottleneck remains the cost of forward/backward substitution; using inner loops in the collision region actually allows us to further lower that cost when this approach is applied.

Comparison with GPU-optimized iterative solver [Komaritzan and Botsch (2019)]

We performed a comparison with iterative solvers that could be natural alternatives to our method, especially for models of more modest resolution and detail. We focused on the GPU-accelerated PCG solver of the recent Fast Projective Skinning technique [Komaritzan and Botsch (2019)] as the most promising recent technique in terms of efficiency and features. Although the highest resolution model in their demos (91K tetrahedra) has 6-7 times fewer tets than our target meshes, they demonstrated real-time performance for models of that scale, making it possible that their technique might scale up to the half-million elements we accommodate. Although we did not have an end-to-end comparison due to differences in collision processing components and their use of embedding vs. conforming meshes in our approach, we performed a study of the comparative convergence efficiency of the two methods in the 500K element regime. Our findings are summarized below. We should however clarify that if one is targeting resolutions below 100K tetrahedral elements, even though both methods would in principle yield interactive performance, the GPU-based PCG solver would not require a prescription of collision regions, and should be preferred due to its generality.

We focused our comparison purely on the solver stage, excluding any runtime cost of assembling or updating the global step matrix or performing collision detection. We also modified their solver [Komaritzan and Botsch (2019)] to include support for embedded collision proxies, which are crucial to our examples. Jacobi preconditioning was used as suggested, although we did not experience any non-trivial acceleration, as our embedding meshes were perfectly regular. We noticed that for high-resolution models the PCG solver required at least 100 (or more) iterations to start approaching the accuracy of the exact solver, and often exhibiting artifacts if inadequate convergence was reached in the global step. As an indication, the cost of 100 iterations (which was the bare minimum for acceptable or even stable convergence) in the GPU PCG solver was 27ms for our *elbow bending* example, and 21ms for our *virtual surgery* scenario. These times are already 2-4x higher than

our inner-loop costs (which produces exact solutions) in instances where multiple inner loops aid convergence, as the elbow simulation. Our method has to sustain the cost of a CPU forward/back-substitution at the beginning/end of each outer PD loop, which takes 70-130ms, but we have experimented with using stock GPU solvers for this stage which typically accelerate this stage by a factor of 3-4x. The direct solver offers the most accurate and robust convergence behavior, does not require parameter tuning to aid convergence (e.g. dynamics leads to much easier matrices for PCG than quasistatics), and is resilient to the stiffness of constraints such as bone attachments and collision penalty forces.

4.6 Conclusions

This chapter presented a localized collision-handling strategy designed to operate efficiently within the Projective Dynamics framework. By employing a Schur complement–based decomposition, the method isolates updates to a collision-prone subset of the simulation domain, enabling the reuse of a prefactored global matrix and preserving interactive performance. Collision proxies are embedded in the mesh and updated in a manner compatible with PD's alternating minimization scheme, maintaining stability even in the presence of dynamic contact events.

The approach is particularly well-suited to scenarios where collisions are spatially localized and persist within a constrained subset of the domain. These conditions are common in a range of volumetric simulation settings, including—but not limited to—anatomically inspired models.

There are, however, important limitations. Because the method is built upon Projective Dynamics, it inherits PD's material model constraints, primarily support for corotated elasticity and compatible variants such as shape targeting. Although our work focuses exclusively on quasistatic simulation, extension to dynamic contexts is conceptually straightforward within the same framework. Frictional contact, on the other hand, remains outside the scope of this work and presents additional challenges for future exploration.

The performance benefits of our approach are most pronounced in large-scale simulations involving hundreds of thousands of elements. In smaller models—one order of magnitude below our demonstrated scale—it may be feasible to perform full matrix refactorizations at each step or rely on iterative solvers without compromising interactivity. The most significant limitation of the current formulation is the assumption that contact regions are relatively small and remain approximately fixed. Many relevant simulation problems do not meet these assumptions and would require more general or adaptive treatments.

Future directions include relaxing the locality constraint on collision regions, incorporating frictional and persistent contact models, and investigating multigrid solvers to accelerate the global step.

5 A COMPUTER BASED FACIAL FLAPS SIMULATOR USING PROJECTIVE DYNAMICS

This chapter demonstrates the stability and computational efficiency of the Projective Dynamics (PD) framework in the context of a realistic facial flap simulator. In addition to extending PD to robustly handle frequent contact events—via the collision-handling strategy introduced in Chapter 6—we further adapt the framework to capture the nonlinear stress–strain behavior of human skin, incorporating material-level nonlinearities beyond those arising from contact alone.

Interactive simulation of surgical procedures has long been a central goal in both computational modeling and medical education, particularly for capturing the mechanical behavior of human skin. Mass-spring models have historically enabled real-time performance but fall short in physical realism, limiting their applicability in surgical planning. In contrast, finite element methods (FEM) offer high fidelity but are often unsuitable for interactive use due to numerical instability and computational cost—especially when simulating large deformations, contact events, or highly nonlinear tissue responses.

To address this gap, this chapter presents a facial flap simulator built on the Projective Dynamics (PD) framework, which provides a practical balance between computational speed and physical stability. By leveraging PD, the simulator can robustly handle rapid and localized nonlinear behavior—such as stiffening at high strain or transient tissue collisions—that previously destabilized our FEM-based approaches. The simulator further incorporates the collision-handling method introduced in Chapter 6, allowing it to maintain interactive performance even in the presence of frequent contact events during surgical manipulation.

The simulator includes a suite of surgical tools designed to reflect common facial reconstruction procedures. These tools support operations such as skin incision, tissue undermining, excision, deep dissection, and retraction using a spring-based skin hook. Sutures may be applied manually or automatically, enabling flexible modeling of layered closure strategies. Using this system, we demonstrate realistic and

interactive simulations of several complex procedures, including Abbe–Estlander lip reconstruction, paramedian forehead flaps, retroauricular flap repairs, and cervicofacial reconstructions.

Together, the use of Projective Dynamics and efficient collision handling provides a foundation for a robust, real-time soft tissue simulation environment. This system supports cognitive surgical training and procedural planning, offering an interactive platform that balances anatomical accuracy with computational performance.

Note on Co-authorship

This chapter is based on joint work with Prof.Eftychios Sifakis, Yutian Tao, Dr. Court Cutting published in *Computer Methods and Programs in Biomedicine* [Wang et al. (2022)]. I was responsible for the algorithm design, implementation, and manuscript preparation.

5.1 Introduction

For several decades, the plastic surgery community has sought effective simulators to support the teaching and rehearsal of local flap closures for facial skin defects. These efforts have followed two primary paths: the construction of high-fidelity physical models and the development of computer-based simulation environments. Among the latter, early systems predominantly utilized mass-spring networks or finite element methods (FEM). While mass-spring models are computationally efficient and easy to implement, they lack the physical accuracy required to simulate the complex behavior of skin under large deformations.

FEM has long been considered the industry standard for simulating solid mechanics, including soft tissue behavior. The first FEM-based facial flap simulation was introduced by Pieper, Laub, and Rosen in 1995 [Berkley et al. (2004)], and since then, numerous offline simulations of flap procedures have been proposed [Berkley et al. (2004); Cardoso et al. (2013); Delingette and Ayache (2004); Kwan

et al. (2020); Lovald et al. (2013); Sifakis et al. (2009); Tang and Wan (2014); Zhang et al. (2017)]. A novel finite element implementation of a flap simulator was developed that enabled real-time performance in a surgical simulation environment [Mitchell et al. (2016, 2015b)]. However, realistic simulation remained a significant challenge due to the highly nonlinear stress–strain response of human skin and the additional nonlinearities introduced by contact and collision processing. These factors severely constrained performance and stability at interactive frame rates.

To address these limitations, Projective Dynamics (PD) was introduced as an alternative numerical framework for simulating deformable bodies in real time [Bouaziz et al. (2014)]. Unlike standard Newton–Raphson solvers that rely on iterative solutions to linearized systems, PD reformulates the simulation as a sequence of weighted least-squares problems. This structure allows for precomputation and efficient updates following changes to the mesh, making PD particularly well-suited for interactive surgical applications. It offers robust and consistent results even in the presence of challenging material behaviors, such as the nonlinear elasticity of skin, and supports stable performance in scenarios involving frequent collisions.

Building on these capabilities, this chapter presents a cognitive facial flap simulator that achieves realistic simulation at interactive frame rates. The simulator enables a range of surgical manipulations, including skin incisions, flap undermining, deep-tissue cutting, retraction using virtual skin hooks, and suture placement. Through the integration of efficient physical modeling and interactive tools, the system supports procedural exploration and training in facial reconstructive surgery.

5.2 Materials and Methods

In this section, we describe the technical components and simulation pipeline of the facial flap simulator. Building on the method introduced in Chapter 4, we detail the modeling process, computational setup, and implementation decisions that support real-time performance during surgical procedures.

The simulator is designed to run on a personal computer equipped with an Intel processor supporting the AVX-512 instruction set (e.g., Xeon Phi or newer), an

NVIDIA GPU (GeForce class or above), and at least 8 GB of RAM. We implemented the simulator in C++ and CUDA, and currently support both Linux and Windows 10 platforms.

To construct the anatomical geometry, we modified a commercial 3D face model using Blender [Blender (2018)] to produce a closed-manifold solid representing facial soft tissue anchored to underlying bone. We developed a suite of surgical tools to support interactive procedures such as skin incisions, flap undermining, and deep-tissue cutting.

To generate a subsurface representation of the skin, we precompute a tetrahedral layer beneath each triangle on the surface mesh. For each triangle, we perform a perpendicular projection to a specified depth, forming a triangular prism composed of three tetrahedra. We then solve this mesh using Projective Dynamics, increasing stiffness for any inverted tetrahedra and repeating the process until a valid, non-inverting volumetric layer is produced. This subsurface is embedded into the overall model. Incisions specified on the skin surface are mirrored in the subsurface to produce full-thickness skin cuts. Undermining operations are defined using a flood-fill algorithm and applied consistently to both surface and subsurface regions to define the flap. For deep cuts, we implement a graph-based tool that interpolates bilinear surfaces between user-defined points and normals.

We embed the full model into a tetrahedral mesh [Molino et al. (2004)], which is updated and submitted to the physics engine after each surgical operation. Using the Projective Dynamics solver described in Chapter 4 [Wang et al. (2021)], we are able to simulate over 500,000 tetrahedra at interactive frame rates., we are able to simulate over 500,000 tetrahedra at interactive frame rates. Forces applied by surgical tools—such as skin hooks and sutures—are implemented as spring constraints and integrated directly into the solver. We tuned the elastic response of the material to match experimental measurements of the nonlinear stress–strain behavior of human skin [Flynn et al. (2013); Joodaki and Panzer (2018); Lapeer et al. (2010); Markenscoff and Yannas (1979); Raposio and Nordström (1998); Ridge and Wright (1965); Rubin and Bodner (2002); Sano et al. (2018); Then et al. (2017); Veronda and Westmann (1970); Weickenmeier et al. (2015)].

Collision handling is supported between key anatomical structures, including contact between the inner lips and teeth, between the eyelids and globes, and between undermined flaps and the exposed tissue bed.

5.3 Results

This section presents a series of simulated procedures to illustrate the capabilities of the facial flap simulator. Figure 5.1 through Figure 5.4 demonstrate various reconstructive surgeries modeled using the tools and methods described above.

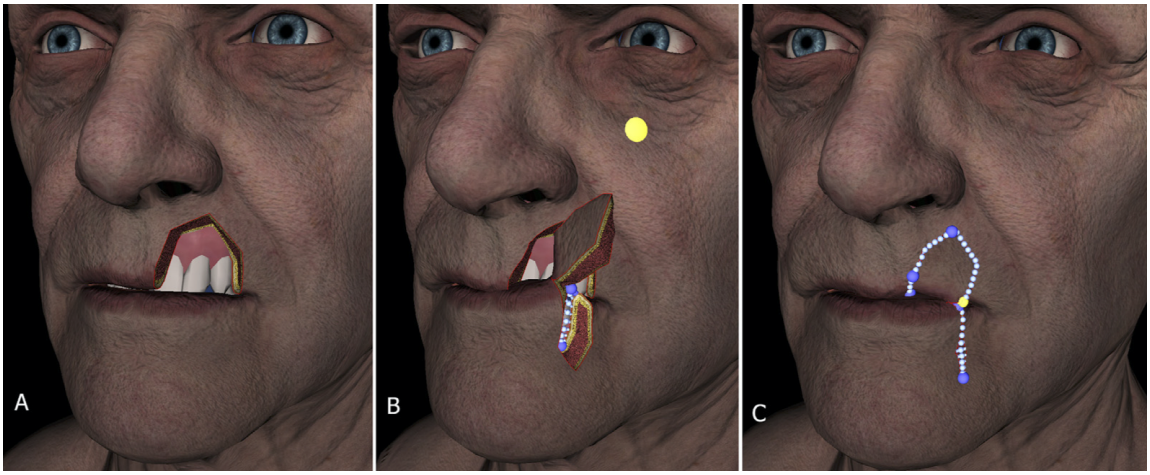

Figure 5.1: Abbe/Estlander resconstruction

Figure 5.1 shows the first stage of an Abbe–Estlander flap reconstruction for an upper lip defect. The procedure is performed using the deep cut tool to create full-thickness incisions through the lip. A small bridge of tissue, preserving the inferior labial artery, is maintained to ensure vascular supply prior to division at a later surgical stage.

In Figure 5.2, a paramedian forehead flap reconstruction is used to close a nasal tip skin defect. The flap is elevated using a combination of skin incision and undermining tools. At the conclusion of the first stage (panel B), the forehead donor site is closed under notable tension, particularly at the superior edge—consistent

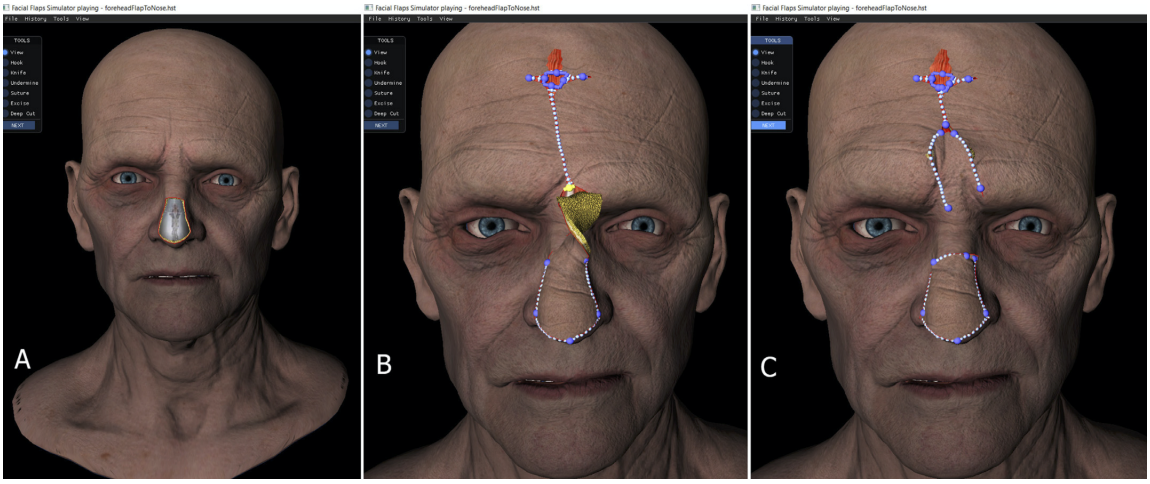

Figure 5.2: Nasal tip skin defect closure

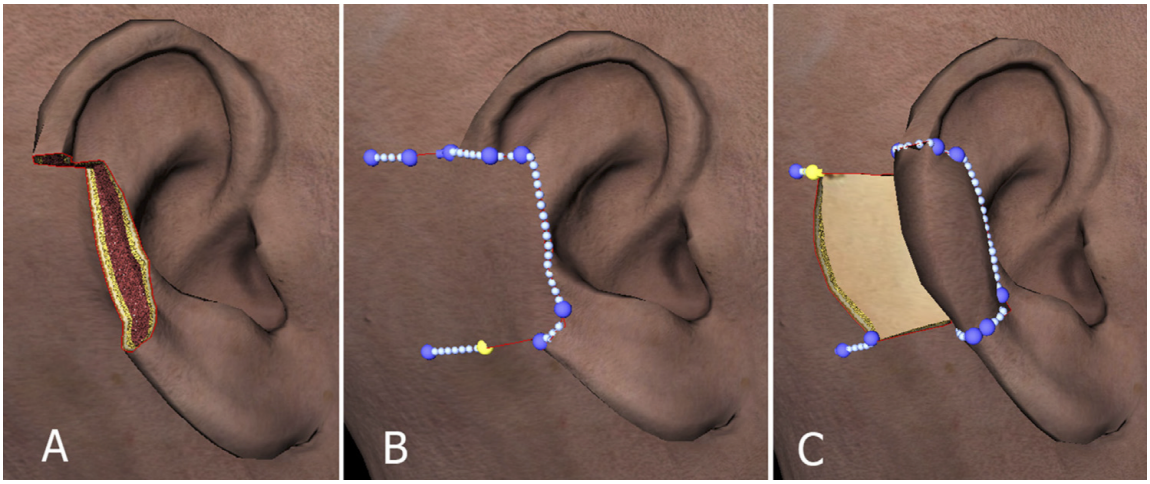

Figure 5.3: Deep cut tool

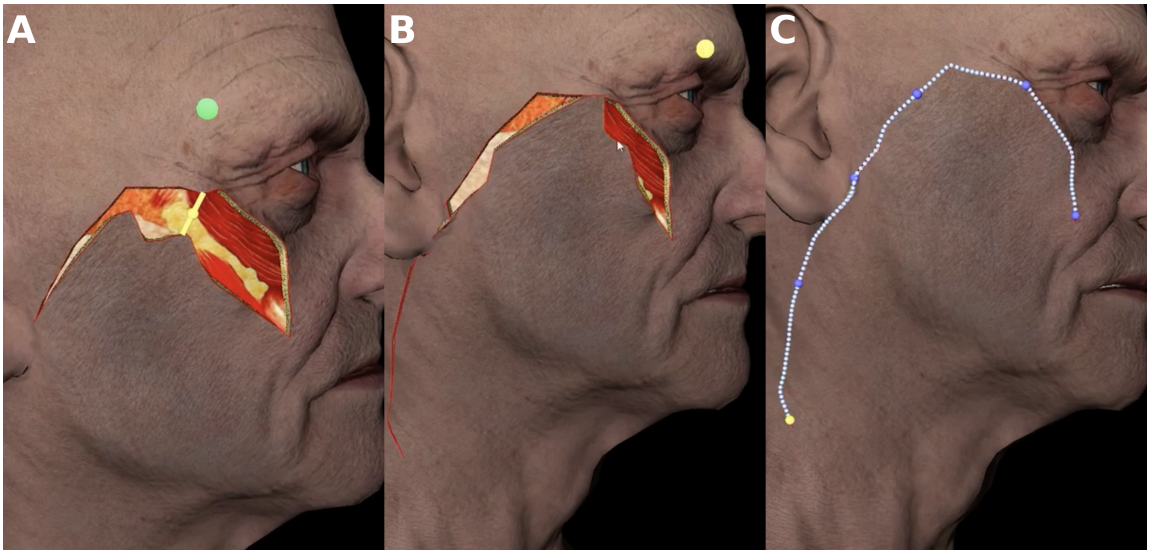

Figure 5.4: Cervicofacial rotation flap

with clinical observations. In the second stage (panel C), the flap is divided, and the remaining pedicle is repositioned into the forehead.

Figure 5.3 presents a reconstruction of a full-thickness auricle defect using a postauricular flap. Panel A shows defect creation with the deep cut tool. In panel B, the flap is transposed and sutured into the anterior defect. In panel C, the flap is divided, wrapped posteriorly, and sutured in place, with the donor site left for secondary coverage or skin grafting.

Figure 5.4 illustrates a cervicofacial rotation flap closure for a cheek defect, highlighting both the nonlinear mechanical response of skin and the interactive surgical toolset. Initially, the flap is unable to reach the defect (Panel A), demonstrating the characteristic resistance of skin under high strain. Following an extended incision and additional undermining (Panel B), the flap becomes sufficiently mobile to close the defect using the suture tool (Panel C).

5.4 Discussion

Physical models have long served as educational tools to illustrate the principles of skin flap design and reconstruction to student surgeons. These models are typically constructed using porcine or cadaveric tissue, or silicone materials mounted on a mechanical substrate [Davies et al. (2013); de Sena et al. (2013); Kazan et al. (2017); Kite et al. (2018); Podolsky et al. (2017); Taylor and Chang (2016); Wu et al. (2015)]. More recently, 3D printing technologies have been employed to fabricate custom anatomical models for surgical simulation, including facial flap reconstructions [Powell et al. (2019)]. Physical models provide a high degree of realism and are often intuitive for the surgical trainee to use. However, they tend to be expensive, single-use, and time-intensive to prepare and reset between sessions.

The tactile component of physical models—valuable in early surgical training—is arguably less critical at the stage of training where flap design and planning become a focus. Psychomotor skills such as incision, undermining, and suturing are generally expected to be well developed prior to plastic surgery training. This reduces the educational advantage of physical models at the cognitive training stage. Additionally, the high refresh rates required to support realistic haptic feedback (typically in the range of 300–1000 Hz) [Hu et al. (2006)] exceed the computational limits of finite element and Projective Dynamics solvers by several orders of magnitude, further limiting the viability of high-fidelity real-time haptics in computational models.

Computer-based simulators can be broadly categorized into two types: psychomotor and cognitive. Psychomotor simulators focus on replicating the physical act of surgery, including visual immersion and tactile feedback. These systems typically employ 3D virtual reality displays and force-feedback devices to replicate the feel of surgical manipulation [Kazan et al. (2017); Satava and Richard (2001); Wallace et al. (2017)]. As a result, they are often seen as digital counterparts to physical models and will likely remain complementary in certain aspects of technical training.

The simulator described in this chapter belongs to the cognitive category. Its

primary goal is to support surgical reasoning and procedural planning rather than to emulate tactile experience. Skin incisions are created by specifying control points on the surface; deep tissue cuts, which require directionality, are defined by additional user-specified vectors. Undermining is performed by connecting incision points along a border line, which propagates to the subsurface. Sutures are modeled as springs connecting either two control points or, for convenience, an automatically generated row of stitches between cardinal sutures. Retraction is simulated using a spring-based "skin hook", shown in Figure 5.5, which applies tension to selected points on the flap boundary.

Figure 5.5: Simulator screenshots of hook tool pulling the skin, showing simulation results of different strain-stress response model; Left: linear response with low stiffness; Middle: biphasic response; Right: linear response with high stiffness.

The primary goal of a cognitive simulator is to reproduce surgical outcomes as accurately as possible. Central to this objective is the realistic modeling of skin and soft tissue responses to surgical forces. Human skin exhibits a highly nonlinear stress–strain relationship [Flynn et al. (2013); Joodaki and Panzer (2018); Lapeer et al. (2010); Markenscoff and Yannas (1979); Raposio and Nordström (1998); Ridge and Wright (1965); Rubin and Bodner (2002); Sano et al. (2018); Then et al. (2017); Veronda and Westmann (1970); Weickenmeier et al. (2015)]. At low strain, skin is highly compliant and deforms easily. However, as strain increases and the collagen network becomes fully extended, resistance to further deformation rises dramatically. In surgical terms, this behavior is colloquially described as "the flap won't reach."

This nonlinear behavior is faithfully reproduced in the simulator described in this chapter. If the skin's stretch limit is exceeded, sutures are unable to close the wound, and retraction forces applied through skin hooks fail to further stretch the tissue. These outcomes reflect realistic surgical challenges and reinforce the importance of capturing physical limits within the simulation environment.

Efforts to model tissue mechanics computationally have a long history. One of the earliest and simplest strategies is the use of mass-spring networks. Montgomery and Schendel developed an early cleft lip simulator based on this approach [Andrea Sorokin and Stephen Schendel (2003); Schendel et al. (2005)]. More recently, mass-spring models have been deployed in mobile applications for basic flap simulation on iOS devices [Naveed et al. (2018)]. Cutting and Oliker also explored mass-spring systems for cleft lip simulation, drawing from prior animation work [Cutting et al. (2002); Oliker and Cutting (2005)]. However, they reported that these models failed to produce sufficiently realistic results. A key limitation of spring-based tetrahedral elements is their inability to resist collapse: as one vertex of a tetrahedron is compressed toward the center of the opposite face, the restoring force diminishes and eventually vanishes, leading to incorrect stable configuration of inversion. This failure mode fundamentally undermines their suitability for modeling biological tissues.

The Finite Element Method (FEM) remains the gold standard for simulating elastic solids [Sifakis and Barbic (2012)]. Pieper, Laub, and Rosen were the first to apply FEM to simulate facial flap procedures in an offline setting [Pieper et al. (1995)]. Subsequent work has extended FEM to simulate a wide variety of flap and breast reconstruction procedures [Berkley et al. (2004); Cardoso et al. (2013); Delingette and Ayache (2004); Kwan et al. (2020); Lovald et al. (2013); Sifakis et al. (2009); Tang and Wan (2014); Zhang et al. (2017); Bielser et al. (2003); Jaberi et al. (2021)]. FEM allows for accurate modeling of nonlinear stress–strain behavior and inversion handling, as described in the work of Irving, Teran, and Fedkiw [Irving et al. (2004)]. However, the computational demands of FEM have historically limited its use to offline applications.

In 2016, advances in solver performance enabled the first real-time FEM-based

flap simulator using a linear elastic model of skin behavior [Mitchell et al. (2016, 2015b)]. Despite this progress, modeling the highly nonlinear elastic response of skin at large strain magnitudes remains a major challenge for interactive simulation. FEM discretizations of the constitutive model of the elastic behavior typically lead to nonlinear systems of equations or equivalent energy minimization problems. Newton–Raphson methods are commonly used to solve these problems but suffer from limited stability guarantees and often rely on heuristic line-search safeguards that compromise convergence speed and accuracy.

These issues are further exacerbated in surgical simulations, where pronounced nonlinearities arise from tissue contact and biphasic material behavior. In such settings, achieving both physical realism and interactive performance demands alternative strategies—such as Projective Dynamics, as explored in this work—that provide improved stability and convergence behavior under complex surgical interactions.

Projective Dynamics (PD), introduced by Bouaziz et al. [Bouaziz et al. (2014)], provides an efficient framework for real-time physics simulation, particularly in the context of deformable body mechanics. PD shares much of its discretization infrastructure with the traditional Finite Element Method (FEM), but replaces the standard Newton–Raphson solver and its iterative linear solver (e.g., conjugate gradients) with a weighted least-squares approach. This reformulation allows the system's Hessian to be approximated as a constant matrix, which can be prefactored and reused across frames, thereby enabling real-time interactivity.

As stated in Chapter 1, a distinguishing advantage of Projective Dynamics is that, when allowed to converge, it reaches the same solution as a standard Newton–Raphson method—yet avoids the need for line search heuristics to ensure stability. This makes PD particularly well-suited for interactive simulations, where early termination is often necessary and robustness is paramount. In the present implementation, we support interactive simulation of over 500,000 embedded tetrahedra on a personal computer [Wang et al. (2021)], with surgical manipulations—such as incisions, suturing, and retraction—triggering recomputation only when the mesh topology changes.

As detailed in Chapter 4, collision handling presents a significant challenge within the Projective Dynamics framework, which relies on a fixed global stiffness matrix to achieve real-time performance. The localized Schur complement–based strategy introduced in the chapter maintains this efficiency by confining computational overhead to a small subset of the mesh. However, this approach inherently limits the scope of collision handling to spatially localized contact events. By restricting updates to anatomically relevant regions—such as the inner lips, eyelids, and the underside of undermined flaps—the simulator is able to incorporate collision response while preserving interactive frame rates in a controlled and efficient manner [Wang et al. (2021)].

The mechanical behavior of skin is complex and varies significantly with patient-specific factors such as age, anatomical site, and individual variability. Skin is anisotropic, viscoelastic, and plastically deformable, and the surgical relevance of each of these properties depends strongly on the timescale of the procedure. For example, mild to moderate anisotropy in facial skin—variation in stiffness along different directions—is of some surgical significance, particularly in planning incisions and closures. Viscoelastic and plastic properties become more important over extended periods—on the order of weeks—as evidenced by the clinical use of tissue expanders to gradually stretch skin for reconstructive purposes [Lee et al. (2021, 2018b)]. For procedures lasting only a few hours, the long-term (plastic) response of skin can largely be ignored, aside from subtle effects such as tissue creep.

Among all mechanical characteristics, the most surgically critical is the point at which skin transitions from easily stretchable to highly resistant. This corresponds to the foot of Region III in the stress–strain curve [Joodaki and Panzer (2018)], where collagen fibers become fully aligned and begin to dominate the mechanical response. This transition threshold varies significantly across individuals and anatomical locations, as demonstrated in both experimental studies [Flynn et al. (2013); Joodaki and Panzer (2018)] and clinical experience. Accurately capturing this nonlinear stiffening response is essential for the realistic simulation of flap reach, tissue tension, and wound closure—key behaviors supported by the simulator presented

in this work.

5.5 Conclusions

This chapter has presented a computer-based cognitive simulator for facial flap surgery, built upon the Projective Dynamics framework. The simulator models nonlinear elastic behavior of skin at interactive frame rates, offering significant advantages over mass-spring networks and traditional Finite Element methods in terms of stability and performance. A targeted collision-handling scheme was incorporated to support anatomically relevant contact scenarios while preserving real-time interactivity. Several representative clinical procedures were simulated, demonstrating the system's ability to capture essential surgical dynamics such as tissue resistance, flap reach, and wound closure.

By combining robust physics modeling with intuitive surgical tools, the simulator provides a flexible platform for cognitive surgical training and planning. It offers a foundation for future extensions involving more complex anatomical sites, patient-specific modeling, and integration with offline high-fidelity solvers.

While this chapter has focused on facial flap simulation, the PD framework has broader implications for surgical modeling. Its performance and stability characteristics make it a promising foundation for more complex procedures involving multiple tissue types and anatomically detailed regions such as the hand, breast, extremities, and thorax. An important future direction is the incorporation of patient-specific geometry and material properties to support personalized surgical planning. Such simulations could be further refined using offline, high-fidelity FEM solvers to evaluate and validate proposed surgical strategies [Lee et al. (2021)]. The combination of fast interactive feedback with offline accuracy evaluation offers a promising path toward clinical-grade simulation tools.

6 LEARNING ACTIVE QUASISTATIC PHYSICS-BASED MODELS FROM DATA

This chapter demonstrates the integration of the Projective Dynamics (PD) framework as a differentiable simulation layer within a learning-based system for modeling internal actuation. In doing so, we extend PD to support both differentiability and active control elements, enabling its use in inverse modeling and learning applications.

Humans and animals can control their bodies to produce a wide range of movements using low-dimensional action signals that represent high-level goals. In this sense, human bodies—and particularly faces—are prime examples of *active* objects: deformable systems capable of reshaping themselves through internal actuation mechanisms. This chapter explores the following proposition (Figure 6.1): given a training set of example poses of an active deformable object, can a differentiable simulator—built upon the Projective Dynamics (PD) framework—enable a learning system to extract a low-dimensional control space that both reproduces the training set and generalizes to novel poses?

Unlike standard machine learning approaches to dimensionality reduction—such as autoencoders—we adopt a physics-based formulation. A differentiable, quasistatic simulation layer based on the PD framework is integrated with a decoder-style neural network, enabling control signals to be learned through physically grounded deformation. This simulator forms the computational backbone of the learning process, allowing the network to optimize control inputs through direct physical feedback. In contrast to anatomical modeling pipelines that rely on imaging data or biological priors, our approach infers both the actuation structure and control behavior directly from example data, without assuming knowledge of underlying musculature. To support high-resolution volumetric models with over 250,000 elements, we introduce a scalable training pipeline that enables efficient backpropagation through the simulation loop.

We demonstrate the system in the context of facial modeling. Trained on a

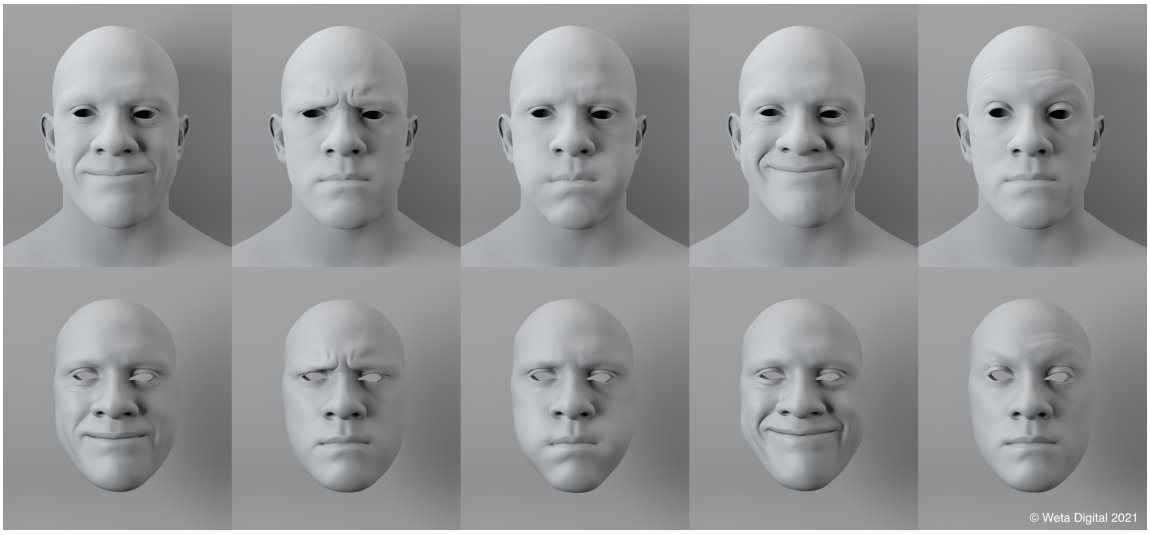

Figure 6.1: This figure illustrates the central question explored in this chapter: given a dataset of target poses for an active deformable model (top row), can a low-dimensional control space be learned to reproduce these poses through a physics-based method (bottom row)? The target configurations are specified as surface meshes (top), and the reconstructed poses are generated using a quasistatic simulator based on the Projective Dynamics (PD) framework operating on tetrahedral volumetric meshes (bottom). Unlike traditional anatomical modeling pipelines that rely on handcrafted musculature models or imaging-derived priors, this approach infers both the internal actuation structure and control parameters directly from data.

corpus of facial expressions, it generalizes to unseen poses, drives high-fidelity animation from sparse motion capture input, and enables intuitive user-guided expression sculpting through the learned control space.

The remainder of this chapter details the update to the Projective Dynamics simulation framework, learning architecture, training procedure, and experimental evaluation, with particular emphasis on the interplay between differentiable physics and data-driven control. By extending PD with differentiability and integrating it into a learning-based pipeline, this work broadens the applicability of PD to inverse modeling and control inference tasks.

Note on Co-authorship

This chapter is based on joint work with Prof.Eftychios Sifakis, Sangeetha Grama Srinivasan, Junior Rojas, Gergely Klár and Prof. Ladislav Kavan published in *ACM Transactions on Graphics* [Srinivasan et al. (2021)]. I contributed significantly to the development of the simulation system, implemented key components of the algorithm, and participated in the experimental evaluation. The chapter has been adapted to highlight and contextualize my contributions.

6.1 Introduction

Many of the elastic deformable bodies used in graphics and visual effects applications are *active* objects, in that their deformation is the result of internal actuation mechanisms in addition to external constraints such as forces, boundary conditions, or collisions. Human bodies and faces are prime examples of such active models, as their apparent deformation is primarily driven by the action of the underlying musculature, which is effectively the built-in actuation mechanism. In graphics, it is common to construct such simulated objects from first principles, by incorporating muscle geometry and composition directly into Finite Element meshes and material models. This chapter explores an alternative approach to constructing and simulating active models from data, asking the following question: Given a collection of shapes that an active object assumes across various scenarios and poses—and assuming they adequately sample its behavioral space—can we infer an actuation mechanism, through a deep learning framework integrated with a physics-based simulator such as Projective Dynamics, that both explains the training examples and generalizes to unseen configurations?

Our approach is built on a central hypothesis: the control parameters underlying an active object's internal actuation mechanism are inherently low-dimensional. This assumption is motivated by anatomical observations—for instance, the apparent shape and motion of human bodies and faces are largely governed by the action of a relatively small set of muscles. In our method, we begin by equipping

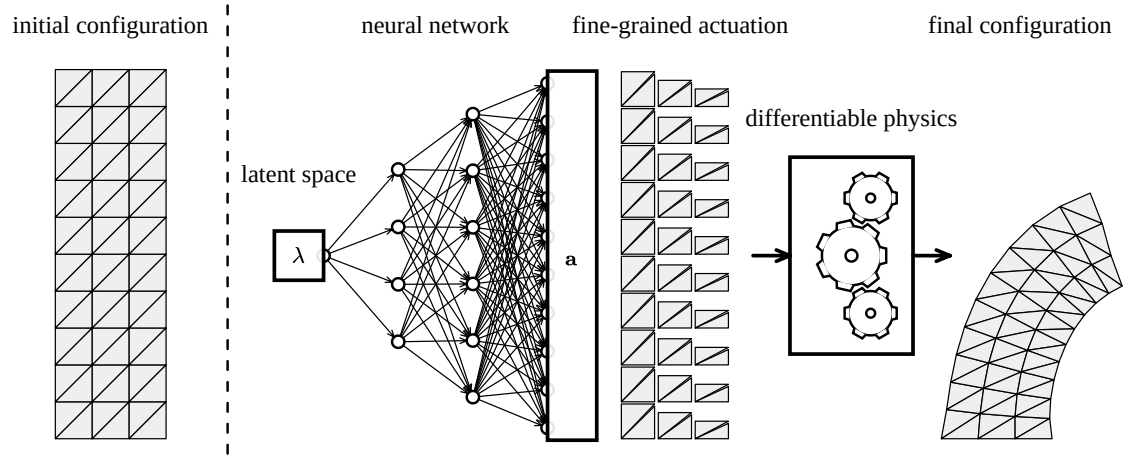

Figure 6.2: Schematic view of our pipeline. We model elastic objects discretized as tetrahedralized elements and endow each element with a fine-grained actuator that can push it towards a prescribed target shape (*"fine-grained actuation"*); a differentiable simulator (*"differentiable physics"*) reconciles the elemental actuation inputs, and produces a quasistatic output shape (*"final configuration"*). We use a decoder-type neural network to generate the fine-grained controls from a low-dimensional latent space (λ) that effectively encodes the structure of the final (coarse-grained) actuators. The network is trained on a collection of final configurations representative of the desired behavior.

the simulated object with highly fine-grained actuation capacity, akin to assigning a dedicated "muscle" to every tetrahedral element in the mesh. This formulation is similar in spirit to prior work on localized muscle control [Ichim et al. (2017); Klár et al. (2020)]. To constrain this otherwise high-dimensional control space, we employ a neural network that maps a compact set of latent control variables to fine-grained actuation signals (Figure 6.2). These signals are then passed to a quasistatic, differentiable Finite Element simulator, which resolves the resulting equilibrium configuration subject to physics constraints such as boundary conditions, external forces, and collisions. After training, the neural network effectively serves as the learned actuation model, encoding how latent control inputs generate observable deformations.

The use of differentiable physics simulators is a rapidly developing area, as

it offers the ability to combine the expressiveness of neural networks with the inductive structure and constraint enforcement of physical models. For example, differentiable simulators based on the material point method (MPM) [Hu et al. (2019, 2020)] have shown promise in soft robotics. However, their reliance on explicit time integration with small time steps can result in substantial memory and computational costs during backpropagation. Implicit methods enable larger time steps and improved stability but are often more numerically demanding in both the forward and backward passes [Hahn et al. (2019); Geilinger et al. (2020)].

Our work builds on a recent extension of Projective Dynamics—originally introduced as an efficient method for forward simulation [Bouaziz et al. (2014)]—into a differentiable framework (DiffPD), which retains PD's fast, stable numerics during both forward and backward passes. This differentiable PD simulator forms the computational backbone of our learning pipeline.

6.2 Related Work

Physics-based modeling and simulation of anatomical systems have been studied for several decades [Weiss et al. (1996); Gladilin (2003); Barbarino et al. (2009); Stavness et al. (2014); Flynn et al. (2015); Zhang et al. (2019)] and usually rely on expert operators who manually design individual organ shapes (bones, muscles, ligaments, etc.), including their connections and material parameters. Several modeling and simulation platforms have been developed, such as FEBio [Maas et al. (2017)], SOFA [Faure et al. (2012)], or ArtiSynth [Lloyd et al. (2012)]. An important feature of physics-based simulation is the ability to faithfully model effects such as contact and volume conservation, which makes a difference in applications such as hand tracking [Smith et al. (2020)]. In this chapter we propose to depart from these traditional modeling paradigms and, instead, we aim to infer muscle models from data using machine learning techniques, specifically, a neural network coupled with a physics-based simulator.

Classical muscle models include the popular Hill-type models [Zajac (1989)], which have been successfully used in both biomechanics [Blemker (2004)] and

computer graphics [Sifakis et al. (2005); Teran et al. (2005a, 2003)]. However, the behavior of muscles *in vivo* is poorly understood mainly due to difficulties of obtaining experimental data [Fan et al. (2014)]. In this project we adopt a shape-targeting approach to muscle modeling [Klár et al. (2020); Nardinocchi and Teresi (2007); Fan et al. (2014); Ichim et al. (2017); Kadlecek and Kavan (2019)]. In this family of models, muscle activations are modeled as volumetric, typically fiber-aligned contractions of the rest pose. Related models were also examined in physics-based animation to generate example-based materials [Martin et al. (2011)] and to control active deformable objects to achieve specific animations [Tan et al. (2012); Coros et al. (2012)]. The key difference of our approach is the ability to learn a low-dimensional action space. Although there is prior work applying dimensionality reduction methods to extract muscle synergies from EMG data [Spüler et al. (2016)] and to obtain control policies for physics-based animation of articulated characters [Ding et al. (2015)], in this chapter we focus on dimensionality reduction via differentiable simulation for active deformable models, with an emphasis on facial animation.

Even though the expressions of the human face are driven by muscles, early methods for facial modeling and animation relied on direct, linear models [Blanz and Vetter (1999)]. Blendshape-based methods dating back to FACS [Ekman and Friesen (1978)] continue to be commonly used in the animation industries [Lewis et al. (2014)]. Local models often provide higher expressivity and ability to generalize to even less common facial shapes [Tena et al. (2011); Wu et al. (2016)]. Deep learning methods enabled a new generation of models that explain both geometry and appearance, in particular "codec avatars" based on variational auto-encoders [Lombardi et al. (2018); Bagautdinov et al. (2018)], including advanced encoders using VR headsets [Wei et al. (2019)] and explicit modeling of the eyes [Schwartz et al. (2020)]. Most recent morphable face models feature impressive visual quality [Li et al. (2020b)] even when learned from just a single scan [Li et al. (2020a)].

Animating the face using physics-based simulation with explicitly modeled muscles is a difficult problem, even though early work on this topic demonstrated

encouraging results [Sifakis et al. (2005)]. About a decade later, the limitations of explicit muscle modeling in targeting 3D facial scans were discussed, along with more expressive muscle models such as "muscle tracks" [Cong et al. (2016)] and fine-grained per-tetrahedron activations [Ichim et al. (2016)]. These advances spurred renewed interest in physics-based animation of the face [Lan et al. (2017); Kozlov et al. (2017)] with more recent approaches proposing inverse modeling combining MRI and 3D scans [Kadlecek and Kavan (2019)]. Related to our approach is a differentiable physics-based model capable of targeting a single image [Bao et al. (2019)], assuming that facial musculature is given. In this chapter, we also propose a differentiable simulator, but we focus on learning the muscle model without making any anatomical assumptions.

We represent our muscle activation model by a neural network, with parameters trained from data. This is related to well-known generative models such as Generative Adversarial Networks [Goodfellow et al. (2014); Arjovsky et al. (2017)] and Variational Auto-encoders [Kingma and Welling (2014); Doersch (2016)], which are generally applicable methods to discover latent structure in data, such as collections of images. The key difference of our methodology is that we explicitly endow our models with knowledge about physics and its numerical solution in the case of quasistatics. Even though physics invariants can be learned from data for simple mechanical systems [Schmidt and Lipson (2009)], the problem becomes more challenging for complex anatomical systems. Our main thesis is that combining neural networks with physics-based priors will improve their training and ability to generalize. Similar questions have recently been studied also for other types of physics-based simulation such as cloth [Geng et al. (2020)].

Differentiable physics-based simulators are an active research area because they enable the combination of the strengths of deep learning frameworks with physics. Differentiable soft-body simulators based on the material point method [Hu et al. (2019, 2020)] have proven useful for training soft robots, but the explicit time integration scheme with small time steps may require a lot of memory and compute time to backpropagate through. Implicit integration enables stable simulation with large time steps, but both the forward and backward passes become much more

complex to solve numerically [Hahn et al. (2019); Geilinger et al. (2020)]. Du et al. (2021) developed concurrently with our work extends Projective Dynamics (originally proposed only for forward simulation [Bouaziz et al. (2014)]) to DiffPD, i.e., differentiable simulator which enjoys the fast numerics of Projective Dynamics also in the backward pass.

6.3 Background

Actuation Model

We adopt shape targeting [Klár et al. (2020)] for our actuation model, which can be defined by extending the conventional definition of corotated energy density function Ψ to depend not only on the deformation gradient $\mathbf{F}$ but also on a shape target matrix $\mathbf{S_t} \in \mathbb{R}^{3\times3}$ for each tetrahedron:

$$\Psi : \mathbb{R}^{3\times3} \times \mathbb{R}^{3\times3} \to \mathbb{R} \tag{6.1}$$

$$\Psi(\mathbf{F}, \mathbf{S}_t) = \arg\min \mathbf{R} \in SO(3)\mu\|\mathbf{F} - \mathbf{R}\mathbf{S}_t\|_F^2 \tag{6.2}$$

It is our intent for $\mathbf{S}_t$ to be a rotationally-invariant descriptor of the shape that an element targets, as opposed to any specific orientation, and the minimization formulation reflects exactly that. With rotations factored away, we restrict $\mathbf{S_t}$ to be a symmetric 3×3 matrix, effectively reducing its degrees of freedom to 6. We represent $\mathbf{S_t}$ as a vector $\mathbf{b} \in \mathbb{R}^6$ using the following convention:

$$\mathbf{S_t} = \begin{pmatrix} 1+\mathbf{b_1} & \mathbf{b_2} & \mathbf{b_3} \\ \mathbf{b_2} & 1+\mathbf{b_4} & \mathbf{b_5} \\ \mathbf{b_3} & \mathbf{b_5} & 1+\mathbf{b_6} \end{pmatrix} \tag{6.3}$$

If we then concatenate the shape targets of all tetrahedra into a single action vector $\mathbf{a} \in \mathbb{R}^{6u}$, where u is the number of tetrahedra, we can reformulate the quasistatic solution χ as a function of $\mathbf{a}$:

$$\chi(\mathbf{a}) = \arg\min \mathbf{x} \in \mathcal{CE}(\mathbf{x}, \mathbf{a}) \tag{6.4}$$

which also implies that the force at the quasistatic equilibrium configuration is zero:

$$\mathbf{f}(\chi(\mathbf{a}), \mathbf{a}) = 0 \tag{6.5}$$

Learning-based Pipeline

Our project focuses on finding tetrahedral actions $\mathbf{a} \in \mathbb{R}^{6u}$ that can reproduce desired poses $\mathbf{t}$ from a data set, specified as vertex positions. Note that because the number of tetrahedra u is typically large, directly controlling $\mathbf{a}$ could be problematic. Instead, we introduce $\mathbf{a} = \pi_\theta(\lambda)$, a mapping from a low-dimensional latent vector $\lambda \in \mathbb{R}^z$ to the high-dimensional action signal $\mathbf{a}$, with learnable parameters θ. We model π_θ using a neural network as shown in Figure 6.3.

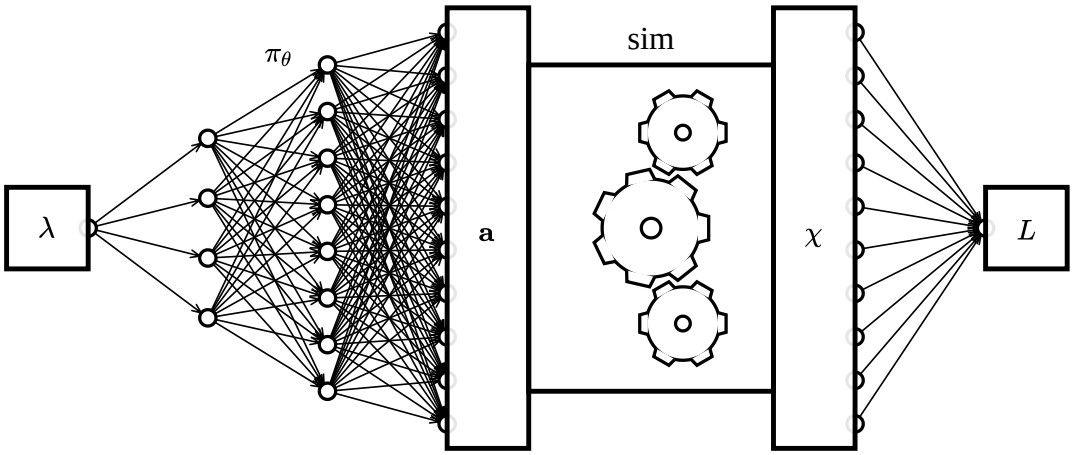

Figure 6.3: Learning-based pipeline. A decoder-type network maps a low-dimensional latent vector λ to a full set of tetrahedral actions $\mathbf{a}$ that are fed to a quasistatic simulator. The simulator produces a resulting pose χ which is used to compute a loss L that measures the discrepancy between the pose produced by the simulator and a given target pose.

Our problem formulation involves finding optimal values for λ and θ to minimize a loss function L that penalizes the discrepancy between poses in the training set $\mathbf{t}_i$ and poses generated by the quasistatic solver $\chi(\mathbf{a}_i)$:

$$\lambda^*, \theta^* = \underset{\lambda,\theta}{\operatorname{argmin}} \sum_i L(\mathbf{a}_i, \mathbf{t}_i) \tag{6.6}$$

$$L(\mathbf{a}_i, \mathbf{t}_i) = \|\chi(\mathbf{a}_i) - \mathbf{t}_i\|^2 \tag{6.7}$$

$$\mathbf{a}_i = \pi_\theta(\lambda_i) \tag{6.8}$$

Note that θ^* is shared among all target poses, since it is a parameter of the mapping π, but λ^* includes one latent vector λ_i^* for each target pose $\mathbf{t}_i$. In other words, if the index i corresponds to time, λ_i expresses temporal variation while the network π_θ expresses spatial variation (by mapping $\lambda_i \in \mathbb{R}^z$ to $\mathbf{a}_i \in \mathbb{R}^{6u}$, where z is much smaller than $6u$).

Back-propagation through Quasistatic Solver

To solve the optimization problem formulated in Equation 6.6 using gradient-based optimization, we need to compute $\frac{dL}{d\mathbf{a}}$. We start by observing that $\frac{d\mathbf{f}}{d\mathbf{a}}$ must be zero when evaluated at $(\chi(\mathbf{a}), \mathbf{a})$ since the force $\mathbf{f}$ is zero at the quasistatic equilibrium configuration as originally stated in Equation 6.5:

$$\frac{d}{d\mathbf{a}}\mathbf{f}(\chi(\mathbf{a}), \mathbf{a}) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\chi(\mathbf{a}),\mathbf{a})} \frac{d\chi}{d\mathbf{a}} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a}}\right|_{(\chi(\mathbf{a}),\mathbf{a})} = 0 \tag{6.9}$$

Using Equation 6.9 we can then obtain a formula for $\frac{dL}{d\mathbf{a}}$ via the following derivations (omitting the explicit evaluations at $(\chi(\mathbf{a}), \mathbf{a})$ for conciseness):

$$\frac{d\chi}{d\mathbf{a}} = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{a}} \tag{6.10}$$

$$\frac{dL}{d\mathbf{a}} = \frac{dL}{d\mathbf{x}}\frac{d\chi}{d\mathbf{a}} = -\frac{dL}{d\mathbf{x}}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{a}} \tag{6.11}$$

As a final step, we take advantage of the fact that the stiffness matrix $-\partial \mathbf{f}/\partial \mathbf{x}$ is *symmetric* to rewrite Equation 6.11 as:

$$\frac{dL}{d\mathbf{a}} = -\underbrace{\left(\underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{-1}}_{\mathbb{R}^{3n\times 3n}} \underbrace{\frac{dL}{d\mathbf{x}}^{T}}_{\mathbb{R}^{3n\times 1}}\right)^{T}}_{\mathbb{R}^{1\times 3n}} \underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{a}}}_{\mathbb{R}^{3n\times 6u}} \tag{6.12}$$

Considering the dimensionality and sparsity of $\frac{\partial \mathbf{f}}{\partial \mathbf{a}}$, this last derivation is particularly useful to make the computation more efficient. Note that $\frac{\partial \mathbf{f}}{\partial \mathbf{a}}$ is sparse because the action associated with each tetrahedral element only affects the force on its four vertices. Although previous works have used similar strategies based on implicit differentiation to compute this derivative [Sifakis et al. (2005); Bern et al. (2019, 2017b,a); McNamara et al. (2004); Wojtan et al. (2006)], the learning-based approach that we propose to discover *low-dimensional* latent spaces for active models with unknown structure via neural networks and differentiable simulation has not been explored before, to our best knowledge.

To put this differentiation strategy in the context of deep learning and back-propagation, we observe that the "forward pass" $\chi(\mathbf{a})$ can be computed by running the physical simulator, and the "backward pass" (which computes $\frac{dL}{d\mathbf{a}}$ assuming that $\frac{dL}{d\mathbf{x}}$ has already been computed) is defined by Equation 6.12 and requires solving a linear system. Note that computing the backward pass is possible because we have access to an explicit definition of the force function $\mathbf{f}$ from our physics-based model. With the forward and backward passes in place, we can compose our physics-based quasistatic simulator with neural networks and run back-propagation through the simulation step.

6.4 Scalability Optimizations

Some of the challenges associated with the effective training of our pipeline are related to the scale of the training set we aspire to accommodate. Specifically, we tar-

get high-resolution volumetric simulation meshes (up to 250K tetrahedral elements in our principal demonstrations), and training sets containing hundreds of frames of sample surface deformations. This section describes the design interventions and algorithmic enhancements that allowed us to accommodate such training tasks on typical well-equipped GPU workstations.

Batch-Optimized Computation

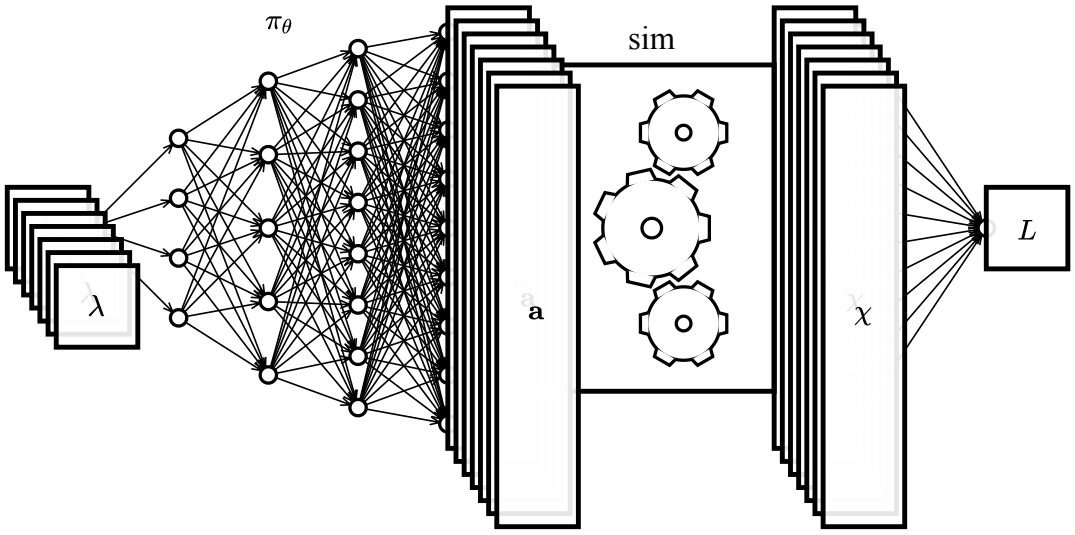

Figure 6.4: Several of the state variables in our training pipeline are provided as batches. State variables in the batch include the latent parameters λ, the fine-grained actions $\mathbf{a}$, and the corresponding result of the quasistatic volumetric simulation χ. Our simulator is designed to support a batch of frames and can accommodate a persistent collection of states for each frame in the batch, thus speeding up training and inference.

We train the neural network on a collection of animation frames corresponding to example surface deformations, with the objective of optimizing the loss function collectively on all frames (Figure 6.4). As a consequence, several of the state variables in our training pipeline are provided as *batches*, with each instance in the batch corresponding to a specific targeted animation frame. Those include the latent

parameters (λ_i), the fine-grained activations ($\mathbf{a}_i$), and the corresponding results of the quasistatic volumetric simulations (χ_i). During training, we conceptually need to maintain several concurrent simulator instances, both for the purposes of forward evaluation (e.g. update of the quasistatic simulation after any changes are made to the latent parameters, during training) as well as back-propagation. Note that, beyond such state variables that are directly referenced in our neural net pipeline, there are intermediate, simulation-related quantities (e.g. Polar Decomposition factors, force accumulators, bone attachment targets and boundary conditions) that are also maintained as a batch across animation frames.

In practice, instead of maintaining several concurrent and independent instances of a simulator, we have adapted a single quasistatic Projective Dynamics simulator engine to accommodate a persistent collection of states corresponding to the various frames of the animation used for training. Individual operations of this engine (e.g. update of elemental rotations via Polar Decomposition, calculation of elastic forces, etc.) can optionally be invoked on a specific frame (or "slice" of the batch), or collectively across all batches, if such in-tandem execution provides performance benefits. We will see that such aggregate execution of certain operations across the entire batch is particularly meaningful in the back-propagation stage, for the evaluation of gradients.

Most operations associated with forward evaluation – corresponding to an update of the quasistatic shape as a result of change in the actuation parameters – are performed on one slice of the batch at a time. This is due to the fact that there are adequate opportunities for efficient parallelization of the relevant simulation kernels by concurrently processing elements of the simulation mesh, that we do not need to resort to processing different animation frames in parallel. This also allows us to iterate the PD loop for as many times as needed for convergence on each individual animation frame, and we do observe that some frames of the animation certainly converge more rapidly than others.

Back-propagation involves substantially different circumstances and operations, which demand more attention to yield good scalability. In particular, the most expensive operation in the pipeline is the inversion of the stiffness matrix $\mathbf{K} = \partial \mathbf{f} / \partial \mathbf{x}$

in equation (6.12) in the process of computing the gradient of the loss with respect to the fine grained actions $\mathbf{a}$. This situation is made more complex by the fact that the stiffness matrix is, in fact, a function $\mathbf{K}_i = \mathbf{K}(\chi_i)$ of the quasistatic shape under the present value of the actions $\mathbf{a}_i$. Essentially, for each instance in the training set we are called upon to independently solve a system $\mathbf{K}_i\mathbf{h}_i = \mathbf{g}_i$ (where $\mathbf{g}_i = \partial L(\chi_i)/\partial \mathbf{x}$) with a different coefficient matrix and right hand side for each frame.

Thankfully, we can leverage an iterative solution, in the spirit of local defect correction schemes, which is similar to the local-global iterative solution in forward simulation of Projective Dynamics. Specifically, if we denote by $\mathbf{K}_{PD}$ the (constant; not deformation-dependent) matrix used in the global step of Projective Dynamics, we can solve the system $\mathbf{K}_i\mathbf{h}_i = \mathbf{g}_i$ using the iterative process:

ALGORITHM 2: Gradient computation using PD matrix

Initialize $\mathbf{h}_i \leftarrow 0$;
`// fixed iteration count`
for $k = 1, 2, \ldots, N$ **do**
  $\mathbf{r} \leftarrow \mathbf{g}_i - \mathbf{K}_i\mathbf{h}_i$; `// matrix-free multiply with` $\mathbf{K}_i$
  $\delta\mathbf{h} \leftarrow \mathbf{K}_{PD}^{-1}\mathbf{r}$; `// via Cholesky factorization`
  $\mathbf{h}_i \leftarrow \mathbf{h}_i + \delta\mathbf{h}$
end

[Du et al. (2021)] provided a good intuitive explanation of this iterative process; its convergence can be rigorously proven, tracing its foundation to the stability properties of the local-global iteration in forward simulation of Projective Dynamics. This methodology alleviates the need to construct or re-factorize every shape-dependent stiffness matrix $\mathbf{K}_i$, as the matrix $\mathbf{K}_{PD}$ can be prefactorized in advance, and the product $\mathbf{K}_i\mathbf{h}_i$ can be computed in a matrix-free fashion, as we do in our simulator (the product is in essence a force differential, which can be computed according to the derivations in [Klár et al. (2020)]. In fact, under this formulation the most computationally expensive part of this solution (and of the back-propagation algorithm, in general) is the solution of $\delta\mathbf{h} \leftarrow \mathbf{K}_{PD}^{-1}\mathbf{r}$ via forward- and back-substitution on a precomputed Cholesky factorization, which needs to be performed for every animation frame in the training set. However, since now the matrix to be inverted

for all frames in a batch has become the same, constant matrix $\mathbf{K}_{PD}$, this task can be recast as a forward- and back-substitution with multiple right hand sides (as many as training frames). This is typically at least an order of magnitude faster than solving each system independently; the substitution operations are highly memory-bound when operating on a single right-hand-side, while using several allows this cost to be amortized, with significant performance gains.

Matrix-Free Multiplication with Force/Action Jacobian

Equation (6.12) involves a multiplication of the vector $\mathbf{h}_i = \mathbf{K}_i^{-1} \frac{\partial L(\chi_i)}{\partial \mathbf{x}}$ (computed using the optimizations of the preceeding section) with the Jacobian $\partial \mathbf{f}(\chi_i, \mathbf{a}_i)/\partial \mathbf{a}$ to form the product $\mathbf{h}_i^T \frac{\partial \mathbf{f}}{\partial \mathbf{a}}$. This is a challenging task in terms of efficiency; the Jacobian $\partial \mathbf{f}/\partial \mathbf{a}$ is a matrix with such large dimensions (remember that $\mathbf{a}$ includes 6 scalars for each tetrahedron) that could only realistically be stored in a sparse format. The assembly of such matrix would be both cumbersome and expensive, and would have to be repeated for each animation frame, as this Jacobian is dependent on the current quasistatic shape, which is different across the batch. All such matrices (which would be very impractical to store) would have to be recomputed at every back-propagation pass.

We circumvent such challenges by computing the product $\mathbf{h}^T \frac{\partial \mathbf{f}}{\partial \mathbf{a}}$ (we drop the animation frame indices i, for simplicity) without explicitly constructing the Jacobian $\partial \mathbf{f}/\partial \mathbf{a}$, but instead applying it to this algebraic operation in a matrix-free fashion. Let us start by revisiting the equation of the Piola stress in our specific actuation model [Klár et al. (2020)]:

$$\mathbf{P}(\mathbf{F}, \mathbf{S}_t) = 2\mu(\mathbf{F} - \mathbf{R}_* \mathbf{S}_t).$$

Where $\mathbf{R}_*$ is the rotational component of $\mathbf{F}\mathbf{S}_t$. We observe that a tetrahedron-specific shape target $\mathbf{S}_t$ (corresponding to six of the degrees of freedom in $\mathbf{a}$, corresponding to the element in question) will only induce stress – and thus, elastic force – on the same element. This the Jacobian $\partial \mathbf{f}/\partial \mathbf{a}$ is conceptually assembled from *elemental* contributions: each will be a sub-matrix which will be non-zero only on the 6

columns corresponding to the actuation values of this element, and the 3×4 rows corresponding to the forces on the four tetrahedron vertices. It is thus sufficient to focus on the contribution of each individual simulation tetrahedron to the product $\mathbf{h}^\top \frac{\partial \mathbf{f}}{\partial \mathbf{a}}$; all such contributions will be additively combined. In doing so, we can also restrict the vector $\mathbf{h}$ to just its values on the four vertices of the element in question.

Thus, the i-th component of the product $\mathbf{h}^\top \frac{\partial \mathbf{f}}{\partial \mathbf{a}}$ is computed as:

$$\mathbf{h}^\top \frac{\partial \mathbf{f}}{\partial \mathbf{a}_i} = \frac{\partial}{\partial \mathbf{a}_i}[\mathbf{h}^\top \mathbf{f}(\mathbf{x}, \mathbf{S}_t(\mathbf{a}))] = \mathbf{h}^\top \delta \mathbf{f}(\mathbf{x}, \mathbf{S}_t; \delta \mathbf{S}_t \leftarrow \frac{\partial \mathbf{S}_t}{\partial \mathbf{a}_i})$$

Here, the derivative $\partial \mathbf{S}_t / \partial \mathbf{a}_i$ – which is constant – is trivially computable from Equation 6.3, while force differentials are directly computable [Sifakis and Barbic (2012)] from the differential of the Piola Stress with respect to a variation $\delta \mathbf{S}_t$ in the shape target. From the definition of $\mathbf{P}$ previously given, this differential is given as:

$$\delta \mathbf{P}(\mathbf{F}, \mathbf{S}_t; \delta \mathbf{S}_t) = -2\mu(\delta \mathbf{R}_*)\mathbf{S}_t - 2\mu \mathbf{R}_*(\delta \mathbf{S}_t)$$

and, finally, the differential of the Polar Decomposition factor $\mathbf{R}_*$ with respect to a variation in $\mathbf{S}_t$ is (following the derivations of [Klár et al. (2020)]):

$$\delta \mathbf{R}_*(\mathbf{F}, \mathbf{S}_t; \delta \mathbf{S}_t) = \mathbf{U}_* \{ \mathcal{E} : [\mathbf{K}^{-1} \mathcal{E}^\top (\mathbf{U}_*^\top \mathbf{F} \delta \mathbf{S}_t \mathbf{V}_*)] \} \mathbf{V}_*^\top$$

where $\mathbf{U}_*, {}^\circ_*, \mathbf{V}_*$ are the SVD factors of the product $\mathbf{F}\mathbf{S}_t$, $\mathcal{E}$ is the alternating tensor, and $\mathbf{K} = \mathrm{tr}({}^\circ_*)\mathbf{I} - {}^\circ_*$ [Klár et al. (2020)]. All the aforementioned calculations are easy to parallelize over the elements of the tetrahedral mesh, share computation among the partial derivatives $\partial / \partial \mathbf{a}_i$ relative to the six components of the elemental action parameter, and can be aggregated to compute the overall product $\mathbf{h}^\top \frac{\partial \mathbf{f}}{\partial \mathbf{a}}$ without explicit formation of the force Jacobian.

6.5 Experiments and Evaluation

We draw our primary set of examples from facial animation, with an additional controlled experiment on simulated, synthetic data on a human limb.

Facial Animation Examples

We draw upon a training set of facial performance data, acquired from an actor that was instructed to articulate their face between several expressive poses. Our training set includes 694 frames, 533 of which are used for training, and 161 frames for testing. The data set we had at our disposal had kinematic information for the mandible, thus boundary conditions associated with skeletal attachments were presumed known at all time. We employ embedded simulation on a background (embedding) simulation mesh constructed from a BCC lattice template, with a total of 246K tetrahedral elements and 52K vertices.

Construction of approximate shape targets We perform a simulation-based extrapolation of the surface data into a volumetric deformation, by targeting the skin surface embedded on the simulation mesh towards the shape provided in the training data (using zero-restlength springs). Approximate shape targets are constructed from the symmetric part of the polar decomposition of the deformation gradient for each simulation tetrahedron. We observed that the use of highly regular, uniform tetrahedra with good aspect ratios was particularly important for our system to have good training and reconstruction performance, hence our choice to use embedding. Prior experiments with conforming meshes with non-ideal aspect ratios would create approximate actions with range beyond what is intutively expected – a way to interpret this is that a flat or sliver tetrahedron might need to scale one of its dimensions very drastically even to assume a shape that is not very remote from its rest shape in absolute distance terms. Our embedded simulation enjoyed very stable performance in this step.

Figure 6.5: Using our differentiable simulator, we can produce a more accurate reconstruction (right) of the target expression (left) than an auto-encoder which only operates on surface shapes and was not trained using the differentiable simulator (middle). The colors indicate reconstruction error, where blue means low error and red means high error.

Autoencoder training and evaluation We then train an autoencoder that maps surface shapes to estimated tetrahedral actions, supervised during training using the elemental shape targets just computed. Even though this network is trained without the intervention of a simulator stage, we can simply take the resulting fine-grain controls and feed them to a simulator, in order to visually assess how close the result is to the surface originally given as input (Figure 6.5). Each epoch of training for the autoencoder takes around 23s on an NVidia Quadro RTX 8000 GPU.

End-to-end training via simulation Using the decoder stage of the autoencoder just trained as initialization, we assemble our decoder-simulator network and continue to train it via backpropagation. Training 533 frames on Intel i9-9940X CPU

(28 cores, 3.30GHz) takes 1603.92s.

Inverse control New poses can be fit against our trained model, by keeping the decoder weights fixed, and optimizing for the latent parameters that yield the best fit. We evaluate this task on our end-to-end trained network, and also contrast to an alternative autoencoder that could be crafted with no simulation (or volumetric deformation) in the loop, that simply performs dimensionality reduction of the animation samples to the same number of parameters as our latent control dimension. We observe our system to be generally superior, and especially so in frames that have mandible motion. It should also be noted that our system provides a full physics-based, simulation-oriented description of the expression, which can be manipulated in additional useful ways, such as incorporating collisions, constraints, volume conservation, etc.

Motion capture and direct manipulation One very practical application of a parametric, controllable face puppet as the one we effectively learn from data would be the generation of detailed facial expressions from sparse motion capture data. We emulate such a scenario by capturing the trajectories of a small set of hand-selected "marker locations" from input shapes that were not part of the training set (effectively, a sparse sampling of unseen data that was also used in our prior reconstruction experiment). We then formulate a loss function that seeks to optimize the sum of squared distances between the target markers and their simulated counterparts, and optimize the latent parameters to compute a fit (Figure 6.7). Our results, illustrate that even a sparse marker set articulates the face model in a way that allows detailed features such as folds, bulges and wrinkles to realistically emerge. Finally, we also demonstrate a "direct manipulation" session, where an artist might directly sculpt a facial expression by dragging a mesh vertex (Figure 6.6), while traversing the parameter space of this puppet (which, in our case, is action- and simulation-driven, as opposed to procedural blendshapes).

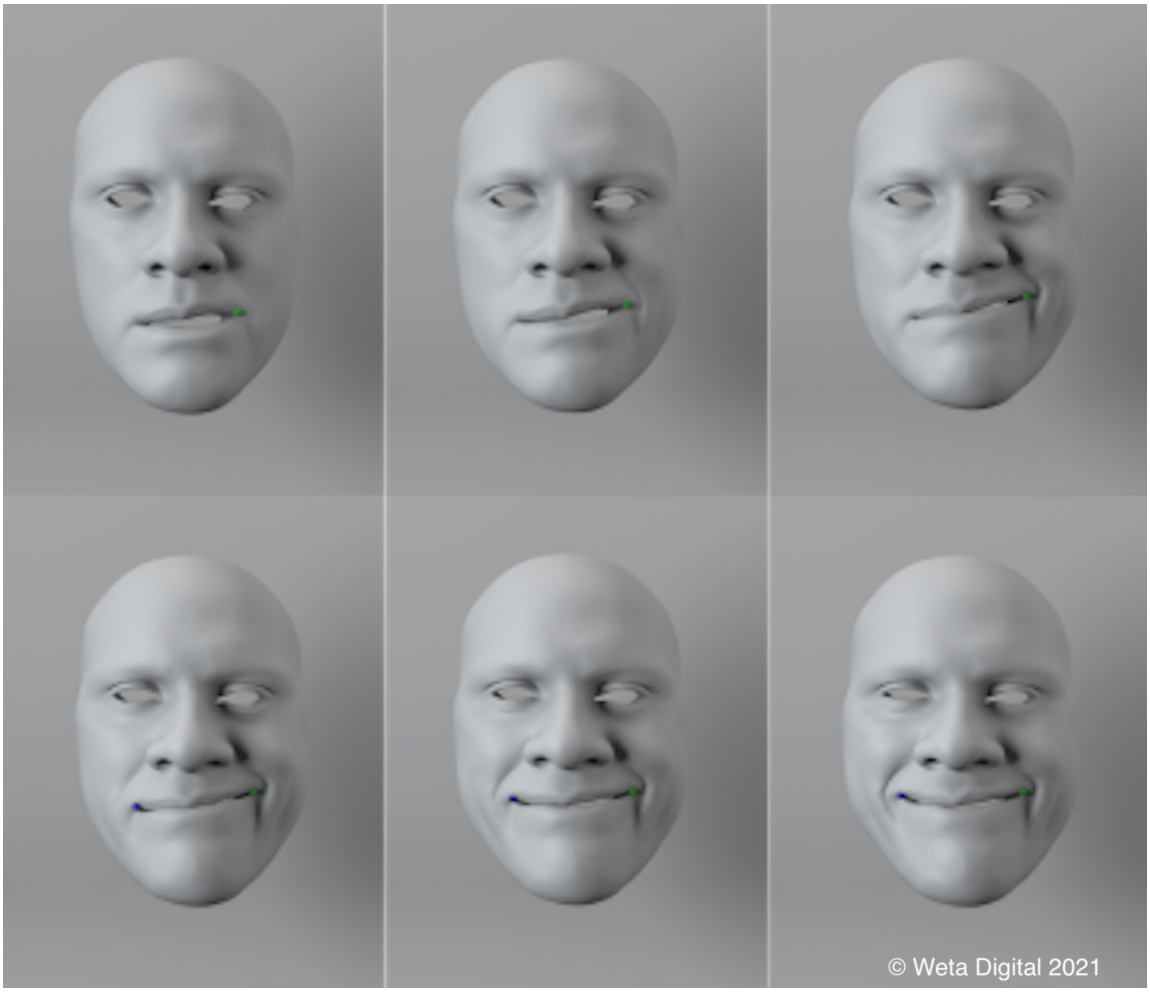

Figure 6.6: Our trained model can be used to generate physically plausible expressions via direct manipulation of sparse targets (blue and green markers). The top 3 images (left-to-right) show results from our pipeline after manipulating the green marker. The bottom 3 images (left-to-right) show the results from our pipeline after manipulating the blue marker, while keeping the green marker fixed.

Learning a Synthetic Elbow Model

The facial dataset we had at our disposal included skeletal motion (for the mandible) to a relatively small degree, and for a smaller fraction of the available frames. We wanted to evaluate the ability of our model, which is action-centric as opposed

Figure 6.7: Motion Capture. One of the applications of our pipeline is to target unseen poses using motion capture markers. The position of the markers (shown on the right) from the unseen pose (left) is given as input to our system. The output from our pipeline is on the right. The red markers correspond to vertices of the surface reconstructed by our embedded simulation and the blue markers correspond to the target positions.

to pose- or shape-centric, to learn the actuation mechanism even in the presence of substantial (albeit, known) skeletal motion. For this specialized example, we created our training set by muscle-driven simulation on the upper extremity; we included four active muscles: the *biceps*, *triceps*, *brachialis*, and *brachioradialis* shown in Figure 6.8. For this experiment we create a tetrahedral embedding mesh 130K elements and approximately 75k degrees of freedom in total. We simulated a sequence where the four muscles activate with a sinusoidal temporal pattern, with different period for each, and also a different period for the kinematic bending of the elbow joint. We repeated the same pipeline as in our facial examples, and demonstrate the performance of our model to fit unseen poses, beyond those in the training set.

6.6 Conclusion

This chapter demonstrates how the Projective Dynamics (PD) framework can be extended to serve as a differentiable simulation layer for learning control models of active deformable objects. By introducing per-element actuation parameters and integrating them into a scalable, quasistatic simulation loop, we enable a learning system to infer internal actuation structures from data without anatomical priors. Our approach leverages a neural network to parameterize a low-dimensional control space and couples it with a differentiable simulator to ensure physical consistency during training.

We presented a full learning pipeline that supports high-resolution volumetric meshes and large animation datasets. The system was evaluated primarily in the context of facial modeling, where it successfully reconstructed and generalized expressive motions from a compact set of latent variables. Additionally, we validated the method on a synthetic arm model to assess its robustness in handling skeletal motion and structured actuation.

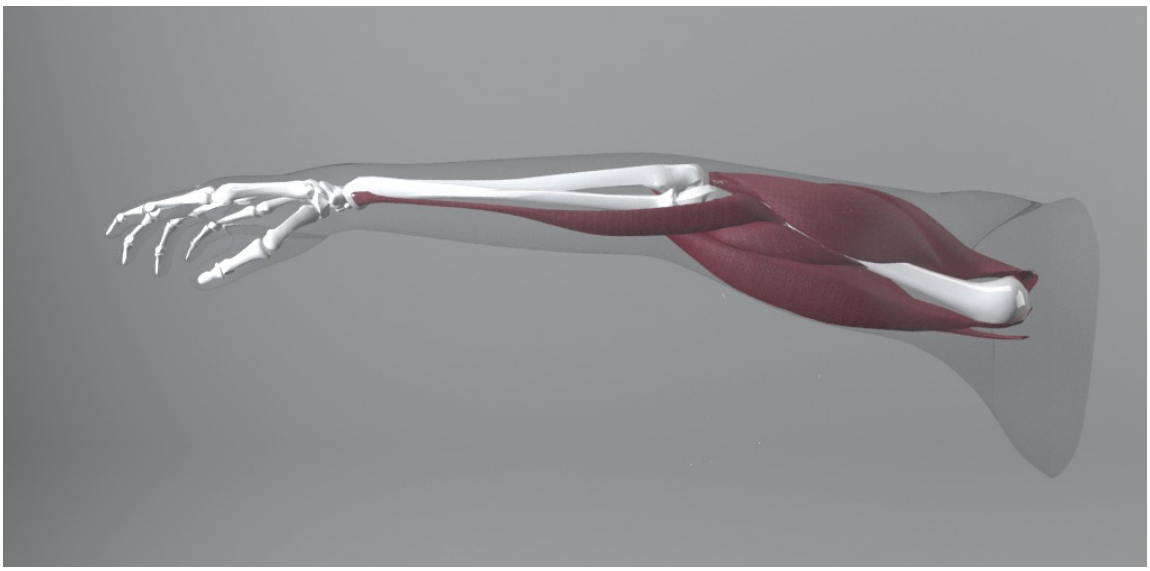

Figure 6.8: Muscle-driven simulation of the arm. We included four active muscles: the *biceps*, *triceps*, *brachialis*, and *brachioradialis*.

The integration of differentiable PD enables backpropagation through physical constraints and material models, providing a principled alternative to purely data-driven shape models. Our formulation allows the latent control parameters to remain interpretable and compatible with downstream applications such as motion capture fitting, interactive expression sculpting, and inverse control.

This work broadens the applicability of PD beyond interactive simulation into data-driven model inference and control learning. The methodology paves the way for future research in learning-based design and control of soft tissues, characters, or soft robots, especially in contexts where ground-truth actuation is unknown or impractical to model directly.

Figure 6.9: Left-to-right. Tetrahedral mesh (column 1) deformed to match the surface poses (column 2). Output form the auto-encoder is shown in column 3. Reconstructions form the end-to-end trained decoder network is shown in column 4. Column 5 shows the result after refining the latent parameters on unseen poses from test data.

7 CONCLUSION AND FUTURE WORK

This dissertation extends the Projective Dynamics (PD) framework to enable efficient, interactive simulation across a broader range of real-world scenarios—including contact-rich environments, nonlinear material behavior, and internal actuation. Across three core technical chapters, we develop scalable algorithms and system-level techniques that retain PD's core strength—its constant global matrix—while expanding its applicability.

7.1 Summary of Contribution

Core Simulator and Collision Handling

We first address the challenge of incorporating contact forces into PD without compromising its efficiency. Traditional penalty-based collision responses alter the simulation energy and break the constancy of the global matrix. We develop a Schur complement method that isolates collision-prone regions, enabling partial factorization of the global matrix and GPU-accelerated solves. The collision-safe bulk of the simulation domain continues to benefit from pre-factored direct solves. We demonstrate this technique in anatomically plausible setups, including deep contact near the elbow and lip-to-teeth collision in cleft palate surgery simulations.

Nonlinear Tissue Behavior and Clinical Application

PD's original formulation assumes linear constitutive behavior, limiting its use in biological tissues that exhibit nonlinear, strain-limiting responses. We address this by designing a high-strain energy term that activates only under large deformation. This term fits into PD's local-global framework, preserving the constancy of the global system matrix while inducing realistic stiffening in overstretched regions. This extension supports robust simulation in facial flap procedures, where skin tissue exhibits biphasic behavior and frequent contact events. We further build

a facial flap simulator that integrates real-time interaction, surgical tool modeling, and topological changes, supporting procedures such as lip, nose, and ear reconstruction.

Internal Actuation and Differentiability

To support control and learning tasks, we extend PD in two directions. First, we incorporate shape-targeting energies to model internal actuation—enabling simulation of muscle-like behaviors using a PD-compatible energy formulation. Second, we introduce a differentiable PD solver based on the adjoint method. By alternating between residual evaluation with the true Hessian and correction steps using the constant PD matrix, we achieve efficient and scalable gradient computation. This differentiable simulator integrates seamlessly with neural network decoders and supports high-resolution tasks such as facial expression reconstruction from motion capture and user-driven expression control, even under altered physical conditions.

7.2 Pending Challenges and Limitations

Though we managed to expand the application of Projective Dynamics with a direct solver to the above scenarios, there are still aspects that need further development.

For our work on collision handling, the major limitation is our conscious assumption that contact regions are relatively small and static. Obviously, there are many examples of highly relevant simulations that would not satisfy such preconditions. For scenarios where global collision is desired, iterative solvers such as Multigrid should be investigated. Although having a constant global matrix does not provide as much acceleration directly towards the solution of the linear system of equations when an iterative solver is used, especially for a matrix-free implementation, the fast direct solver with the global matrix can potentially be used as a preconditioner for the iterative solver to accelerate its convergence.

On simulating the biphasic behavior of skin, we achieve strain-limiting behavior by superimposing a Projective Dynamics-compatible energy term that only induces

forces when operating under high strain. This way, the efficiency and stability guarantee of Projective Dynamics is maintained through a constant positive definite global matrix. However, for regions of the simulation mesh operating under strain lower than the strain limit, the positive definite matrix superimposed due to the addition of the strain-limiting energy deviates the global matrix further away from the true hessian of the total energy function, resulting in slow convergence of PD iterations.

In the facial expression project, the use of shape targeting [Ichim et al. (2017); Klár et al. (2020)] as the underlying fine-grained actuation model was motivated by its stability properties and the moderate nonlinearity it involves. It has also been argued [Klár et al. (2020)] that it is a plausible generalization of fiber-based muscle actuation models. It does come, however, with certain limitations; the principal among them is that the "action" of the contractile elements principally dictates their desired shape, not their stiffness. For certain parts of anatomical models, the distinction is substantial: tendons are characterized by their highly anisotropic stiffness, and modeling them as such is essential to their biomechanical function (including their ability to generate adequate torque to move the skeleton). Since our learning objective is primarily geared towards creating shapes, it is questionable if our approach would be directly applicable to a musculoskeletal application where the primary function would be supporting skeletal motion (as opposed to deforming the skin). Therefore, it would be a fascinating topic of inquiry to explore learning-based models that could also attenuate their stiffness and infer such properties from data.

Also, our examples in this iteration of our work do not include collision effects as a component of the training process or the refinement process that determines the values of the latent parameters that best fit an unseen face shape. We do include some demonstrations of collision processing as a post-process (while replaying precomputed elemental actions), but not as a part of our optimization pipeline. We do, in fact, suffer some isolated consequences of this omission. For example, the absence of lip/gum collisions in the jaw makes it difficult to have a good match with the animation data in scenarios where the jaw is widely open. Although this

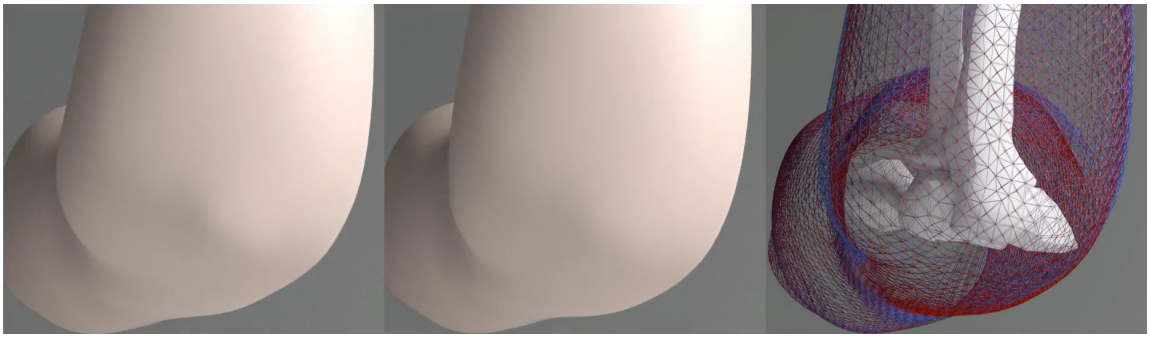

Figure 7.1: We compare the simulation result of the arm-bending example with and without the volume preservation term in corotational elasticity. Left: simulation result with $\lambda = 1$; Middle: simulation result with $\lambda \neq 0$; Right: wireframe comparison of the two results, with blue one with the term, and the red one without.

is a feature presently not incorporated into our pipeline, we strongly believe that our physics-based approach creates much more favorable conditions for training models through scenarios involving collision and reconstructing expressions (*e.g.*, via motion capture) while respecting collisions. Additionally, since we have seen examples of inverse control in the presence of self- and body-collisions [Sifakis et al. (2005)], it is reasonable to expect that tuning the latent parameters of a pre-learned mechanism to target poses through collisions should be quite possible.

Last but not least, using Projective Dynamics naturally limits the supported material models (e.g., corotated elasticity, shape targeting), and thus are confined to scenarios where such models are suitable as constitutive models. For the second term in corotational elasticity, $\frac{\lambda}{2}\mathrm{tr}^2(R^T F - I)$, penalizing deviation of the sum of deformation in the principal directions from three (total principal deformation of identity matrix), aims to preserve volumes of elements. However, as the energy term does not describe a quadratic distance from the degree of freedoms to a constraint manifold (multiplying by R^T brings in nonlinearities), there is no direct application of the Projective Dynamics paradigm to handle the volume preservation constraint as dictated in corotational elasticity. Compared to the energy term used in this document, the simulation result from using the entire corotational energy formulation does show additional resistance to volume change (as illustrated in

figure 7.1). However, we note that in corotational energy, it is not the true volume preservation constraint ($\det(F) = 0$) that is being enforced. Therefore, corotated elasticity may not be a suitable constitutive model for simulation scenarios where volume preservation needs to be enforced strictly. For the muscle actuation example shown in our visual demonstration, although the characteristic bulging effect of muscles results from contraction under preserved muscle volume in the real world, we compensate this with target shapes that have unchanged volumes ($\det(S) = 1$), pushing the muscles to adopt a bulging configuration.

7.3 Future Work

Convergence of Strain Limiting Energy

Our remedy for the strain-limiting effect results in deteriorated progression of the PD global iterations. This slow convergence of global steps, in turn, leads to slow progression in the PD local steps, where rotation projections are performed. Overall, in an interactive setting, this inadequate convergence manifests as additional resistance to rotation. This adverse effect is, in fact, observed in our facial flaps simulator and is especially prominent in the presence of thin topological structures where bending and twisting become significant modes of deformation. Thus boosting convergence speed for Projective Dynamics with strain-limiting elements is essential to improve the realism of the facial flaps simulator.

One observation on strain distribution of passive simulation objects is that elements under high strain are concentrated near the region where boundary constraints are enforced. One venue to pursue is to accelerate Projective Dynamics for biphasic-behaving skin through the prefactorization of multiple global PD matrices, each corresponding to the activation of high-strain energy in carefully designed groups of elements. At each PD iteration, the global matrix with the tightest covering set of high-strain elements is chosen as the global direct solve. Performance-wise, we notice that all these global PD matrices have the same sparsity pattern. Therefore the additional cost for numerical factorizing these in parallel

would be relatively small compared to that of a full Cholesky factorization, where symbolic factorization exploiting parallelization opportunity in sparsity structure take the majority of computational time.

Learning Contact Inflicted Frames

As stated above, collision is not taken into account by the training process of the system. The problem with this is that, unaware of the collision mechanism, the learning pipeline will attribute such deformation to the actuation of the element when there are collision-inflicted regions in the training data. Therefore, this unawareness of collision will yield artifacts when generalizing learned models to unseen poses. Nonetheless, including collision in gradient computation for the backward pass is not trivial due to the highly non-linear nature of the collision response. However, this would be a fascinating research opportunity that would unlock significant new capabilities.

A naive adoption of the Schur complement technique from the forward stage might amortize the computational cost of full matrix refactorization. However, volatility of collision location across different learning frames would result in different global matrix being solved for each frame, impairing the simulator's ability to perform the batched linear solve efficiently with multiple right-hand sides with the same, constant, pre-factorized matrix when evaluating the gradients for back-propagation. Therefore, to expand our learning pipeline to deal with training data with collision, we would like to investigate the remedy where, keeping the essence of Algorithm 2, we perform the direct solve step using a global matrix with all collision assumed to be active, and, again, compute residual using the true hessian where only active collision constraints are being considered.

7.4 Closing Remarks

The extensions to Projective Dynamics presented in this dissertation aim to strike a careful balance between physical realism and computational efficiency—two

objectives often at odds in the simulation of deformable bodies. While this work focused on direct solvers and specific material models, the techniques and principles developed here open up many directions for future systems, including those based on iterative solvers, more expressive constitutive models, and hybrid data-driven approaches.

Ultimately, I hope this work contributes not only to the practical deployment of simulation in graphics, medicine, and machine learning, but also to a deeper understanding of how structure-aware numerical methods can scale to real-world complexity while preserving interactivity, differentiability, and robustness.

A PROOF THAT THE GLOBAL MATRIX IS BLOCK DIAGONAL

As stated above, the global step energy is

$$E(\vec{x}) = \sum_e \mathrm{Vol}_e \|F_e(\vec{x}) - R_e\|_F^2 \tag{A.1}$$

Let us first consider contribution to global stiffness matrix in an element. For the simplicity of the notification, we drop the subscript e with the understanding that all quantities below F, R, Ψ, etc. are computed in a single element. Given elemental displacements $D_x = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \end{bmatrix}$, we have deformation gradient

$$F = D_x \begin{bmatrix} -1^T \\ \mathbb{I} \end{bmatrix} D_m^{-1} = D_x G^T$$

We can compute elemental force $H = \begin{bmatrix} f_0 & f_1 & f_2 & f_3 \end{bmatrix}$, where $f_i, i = 0, 1, 2, 3$ are the forces on the nodes of the tetrahedral element arising from the element. As

$$E_e = \mathrm{Vol}\|F - R\|_F^2 = \|D_x G^T - R\| \tag{A.2}$$

Then the entries in the element stiffness matrix (a 4th order tensor) can be computed by

$$\frac{\partial^2 \Psi_e}{\partial x_i^{(j)} \partial x_k^{(l)}} = \frac{\partial}{\partial x_i^{(j)}}(-H_{lk}) = -(\frac{\partial H}{\partial D_x})_{ji} = e_j^T (\frac{\partial H}{\partial D_x}) e_i \tag{A.3}$$

And as elemental force can be computed through first Piola stress tensor: $H = -\mathrm{Vol} P G$. The entries in H can be written as $H_{lk} = -\mathrm{Vol} e_l^T P G e_k$. Taking the differ-

entials, we have

$$\delta H_{lk} = -\text{Vol} e_l^T \delta P G e_k \tag{A.4}$$
$$= -\mu \text{Vol} e_l^T \delta D_x G^T G e_k \tag{A.5}$$
$$= -\mu \text{Vol} \text{tr}(e_l^T \delta D_x G^T G e_k) \tag{A.6}$$
$$= -\mu \text{Vol} \text{tr}(G^T G e_k e_l^T \delta D_x) \tag{A.7}$$
$$= -\mu \text{Vol}(e_l e_k^T G^T G) : \delta D_x \tag{A.8}$$
$$\Leftrightarrow \frac{\partial H}{\partial D_x} = e_l e_k^T G^T G \tag{A.9}$$

Plug Equation A.9 in Equation A.3, we obtain

$$\frac{\partial^2 \Psi_e}{\partial x_i^{(j)} \partial x_k^{(l)}} = 2\mu \text{Vol} e_j^T e_l e_k^T G^T G e_i \tag{A.10}$$
$$= 2\mu \text{Vol} \delta_{jl} e_k^T G^T G e_i \tag{A.11}$$

It follows that for diagonal blocks of the elemntal stiffness matrix, $\frac{\partial^2 E}{\partial \vec{x}^{(i)^2}} = \sum_e \text{Vol}_e G_e^T G_e, \forall i$ are identical, and that the off-diagonal blocks, $\frac{\partial^2 E}{\partial \vec{x}^{(i)} \partial \vec{x}^{(j)}} = 0, \forall i \neq j$. *i.e.*, global step stiffness matrix K_{el} (when $\lambda = 0$) is block diagonal with three identical diagonal blocks.

B DERIVATION ON INNER LOOP COMPUTATION

As stated in Equation 4.17, with auxiliary variables R_α, R_β computed and fixed from the local step, the global-step energy has the form

$$E(\vec{x}) = \hat{E}_\alpha(x_1, x_2; R_\alpha) + \hat{E}_\beta(x_2; R_\beta) + E_{col}(x_2) \tag{B.1}$$

As each of the energy terms is of a pure quadratic form without loss of generality, let us assume

$$\hat{E}_\alpha(x_1, x_2; R_\alpha) = \frac{1}{2}x_1^T K_{11} x_1 + x_1^T K_{12} x_2 + \frac{1}{2}x_2^T K_{22}^\alpha x_2 + b_{1;\alpha}^T x_1 + b_{2;\alpha}^T x_2 + c_\alpha \tag{B.2}$$

$$\hat{E}_\beta(x_2; R_\beta) = \frac{1}{2}x_2^T K_{22}^\beta x_2 + b_{2;\beta}^T x_2 + c_\beta \tag{B.3}$$

$$E_{col}(x_2) = \frac{1}{2}x_2^T C x_2 + d^T x_2 + c_c \tag{B.4}$$

Let us emphasize that the first-order terms (the b terms) in the quadratic form depend on the corresponding values of R's computed in local steps and thus vary for each PD iteration. We apply Newton Raphson method to minimize the pure quadratic energy, which will reach convergence in just one newton's iteration:

$$K_{11}\delta x_1 + K_{12}\delta x_2 = f_{1;\alpha}(x_1, x_2) \tag{B.5}$$

$$K_{21}\delta x_1 + K_{22}\delta x_2 + C(x_2)\delta x_2 = f_{2;\alpha}(x_1, x_2) + f_{2;\beta}(x_2) + f_c(x_2) \tag{B.6}$$

Where

$$f_{1;\alpha}(x_1, x_2) = -\frac{\partial \hat{E}_\alpha}{\partial x_1} = -K_{11}x_1 - K_{12}x_2 - b_{1;\alpha} \tag{B.7}$$

$$f_{2;\alpha}(x_1, x_2) = -\frac{\partial \hat{E}_\alpha}{\partial x_2} = -K_{22}^{\alpha}x_2 - K_{21}x_1 - b_{2;\alpha} \tag{B.8}$$

$$f_{2;\beta}(x_2) = -\frac{\partial \hat{E}_\beta}{\partial x_2} = -K_{22}^{\beta}x_2 - b_{2;\beta} \tag{B.9}$$

$$f_c(x_2) = -\frac{\partial E_{col}}{\partial x_2} = -Cx_2 - d \tag{B.10}$$

can be interpreted as forces due to each energy component. Performing elimination with respect to Schur decomposition

$$\delta x_1 = K_{11}^{-1}(f_{1;\alpha}(x_1) - K_{12}\delta x_2) \tag{B.11}$$

Plug B.11 into B.6, we get

$$K_{21}(K_{11}^{-1}(f_{1;\alpha} - K_{12}\delta x_2)) + K_{22}\delta x_2 + C\delta x_2 = f_{2;\alpha}(x_1, x_2) + f_{2;\beta}(x_2) + f_c(x_2) \tag{B.12}$$

$$\Leftrightarrow (K_{22} - K_{21}K_{11}^{-1}K_{12})\delta x_2 = f_{2;\alpha}(x_1, x_2) + f_{2;\beta}(x_2) + f_c(x_2) - K_{21}K_{11}^{-1}f_{1;\alpha}(x_1) \tag{B.13}$$

At first glance, to compute the right-hand side above, x_1 needs to be re-evaluated each time, and the four terms in the right-hand side can then be computed, including one that involves a solve with K_{11}, we will quickly show below that this is not the case. Let us compute

$$\begin{aligned} g(x_1, x_2; R_\alpha) &= f_{2;\alpha}(x_1, x_2) - K_{21}K_{11}^{-1}f_{1;\alpha}(x_1) \\ &= -K_{22}^{\alpha}x_2 - K_{21}x_1 - b_{2;\alpha} + K_{21}K_{11}^{-1}(K_{11}x_1 + K_{12}x_2 + b_{1;\alpha}) \\ &= -(K_{22}^{\alpha} - K_{21}K_{11}^{-1}K_{12})x_2 - (b_{2;\alpha} - K_{21}K_{11}^{-1}b_{1;\alpha}) \end{aligned} \tag{B.14}$$

If we fix the auxiliary variables R_α, re-computation of x_1 is not necessary for evaluating the right-hand side above, resulting in local solving iterations that only

recompute x_2 and R_β. If we look further into the computation of the terms, we have

$$g(x_1, x_2; R_\alpha) = -\Sigma_\alpha(x_2 - x_2^*) - \Sigma_\alpha x_2^* - (b_{2;\alpha} - K_{21}K_{11}^{-1}b_{1;\alpha}) \tag{B.15}$$

$$= f_{2;\alpha}(x_1^*, x_2^*) - K_{21}K_{11}^{-1}f_{1;\alpha}(x_1^*) - \Sigma_\alpha(x_2 - x_2^*) \tag{B.16}$$

$$= g(x_1^*, x_2^*; R_\alpha) - \Sigma_\alpha(x_2 - x_2^*) \tag{B.17}$$

Where $\Sigma_\alpha = K_{22}^\alpha - K_2 1 K_{11}^{-1}K_{12} = \Sigma - K_{22}^\beta$ is the schur complement of the stifffness matrix K when only the influences of energy term E_α is considered. From equation B.17, we can see that given $g(\cdot)$ at any x_1^*, x_2^*, we can compute $g(x_1, x_2)$ with the additional computation of $\Sigma_\alpha(x_2 - x_2^*)$. Here, we also notice that $g(x_1^*, x_2^*)$ can be obtained by the partial forward substitution resulting from the Schur complement system:

$$\begin{bmatrix} L_1 & \\ K_{21}L_1^{-T} & I \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} f_{1;\alpha}(x_1^*, x_2^*) \\ f_{2;\alpha}(x_2^*) \end{bmatrix} \tag{B.18}$$

$$\Rightarrow y_1 = L_1^{-1}f_{1;\alpha}(x_1^*, x_2^*) y_2 = g(x_1^*, x_2^*) \tag{B.19}$$

Choosing (x_1^*, x_2^*) as the results obtained from the previous global-local step, we effectively harvest the computation in forward substitution and rewrite Equation B.13 as

$$(K_{22} - K_{21}K_{11}^{-1}K_{12})\delta x_2 = f_{2;\beta}(x_2) + f_c(x_2) + g(x_1^*, x_2^*) - \Sigma_\alpha(x_2 - x_2^*) \tag{B.20}$$

REFERENCES

Abu Rumman, Nadine, and Marco Fratarcangeli. 2015. Position-based skinning for soft articulated characters. *Comput. Graph. Forum* 34(6):240–250.

Andrea Sorokin, Giancarlo Lionetti', and MD Stephen Schendel. 2003. A surgical simulator for cleft lip planning and repair. *Medicine Meets Virtual Reality 11: NextMed: Health Horizon* 94(8):204.

Angles, Baptiste, Daniel Rebain, Miles Macklin, Brian Wyvill, Loic Barthe, JP Lewis, Javier Von Der Pahlen, Shahram Izadi, Julien Valentin, Sofien Bouaziz, et al. 2019. Viper: Volume invariant position-based elastic rods. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2(2):1–26.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223.

Bagautdinov, Timur, Chenglei Wu, Jason Saragih, Pascal Fua, and Yaser Sheikh. 2018. Modeling facial geometry using compositional vaes. In *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)*.

Bao, Michael, Matthew Cong, Stephane Grabli, and Ronald Fedkiw. 2019. High-quality face capture using anatomical muscles. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr)*.

Barbarino, GG, M Jabareen, J Trzewik, A Nkengne, G Stamatas, and E Mazza. 2009. Development and validation of a three-dimensional finite element model of the face. *Journal of biomechanical engineering* 131(4):041006.

Bender, Jan, Matthias Muller, Miguel A. Otaudy, and Mattias Teschner. 2013. Position-based Methods for the Simulation of Solid Objects in Computer Graphics. In *Eurographics (state of the art reports)*, 1–22.

Berkley, Jeffrey, George Turkiyyah, Daniel Berg, Mark Ganter, and Suzanne Weghorst. 2004. Real-time finite element modeling for surgery simulation: An

application to virtual suturing. *IEEE Transactions on visualization and computer graphics* 10(3):314–325.

Bern, James, Grace Kumagai, and Stelian Coros. 2017a. Fabrication, modeling, and control of plush robots. In *2017 ieee/rsj international conference on intelligent robots and systems (iros)*, 3739 – 3746. IEEE. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017); Conference Location: Vancouver, Canada; Conference Date: September 24-28, 2017.

Bern, James M, Pol Banzet, Roi Poranne, and Stelian Coros. 2019. Trajectory optimization for cable-driven soft robot locomotion. *Proceedings of Robotics: Science and Systems*.

Bern, James M., Kai-Hung Chang, and Stelian Coros. 2017b. Interactive design of animated plushies. *ACM Trans. Graph.* 36(4):80:1–80:11.

Bielser, Daniel, Pascal Glardon, Matthias Teschner, and Markus Gross. 2003. A state machine for real-time cutting of tetrahedral meshes. In *11th pacific conference oncomputer graphics and applications, 2003. proceedings.*, 377–386. IEEE.

Blanz, Volker, and Thomas Vetter. 1999. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on computer graphics and interactive techniques*, 187–194.

Blemker, Silvia Salinas. 2004. 3d modeling of complex muscle architecture and geometry. Ph.D. thesis, Stanford University.

Blender, O. 2018. Blender—a 3d modelling and rendering package. *Retrieved. represents the sequence of Constructs1 to* 4.

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

Brandt, Christopher, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-Reduced Projective Dynamics. *ACM Transactions on Graphics* 37(4):1–13.

Cardoso, Augusta, Gustavo Coelho, Horácio Zenha, Vera Sá, Georgi Smirnov, and Horácio Costa. 2013. Computer simulation of breast reduction surgery. *Aesthetic plastic surgery* 37(1):68–76.

Chao, Isaac, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. *ACM transactions on graphics (TOG)* 29(4):38.

Cong, Matthew, Kiran S Bhat, and Ronald Fedkiw. 2016. Art-Directed Muscle Simulation for High-End Facial Animation. In *Proceedings of the acm siggraph/eurographics symposium on computer animation - sca '16*, 119–127.

Coros, Stelian, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. 2012. Deformable objects alive! *ACM Transactions on Graphics (TOG)* 31(4):69.

Cutting, C, A Oliker, J Haring, J Dayan, and D Smith. 2002. Use of three-dimensional computer graphic animation to illustrate cleft lip and palate surgery. *Computer Aided Surgery* 7(6):326–331.

Davies, Jennifer, Manaf Khatib, and Fernando Bello. 2013. Open surgical simulation—a review. *Journal of surgical education* 70(5):618–627.

Daviet, Gilles. 2020. Simple and scalable frictional contacts for thin nodal objects. *ACM Trans. Graph.* 39(4).

Delingette, Hervé, and Nicholas Ayache. 2004. Soft tissue modeling for surgery simulation. *Handbook of Numerical Analysis* 12:453–550.

Dinev, Dimitar, Tiantian Liu, and Ladislav Kavan. 2018a. Stabilizing integrators for real-time physics. *ACM Transactions on Graphics* 37(1).

Dinev, Dimitar, Tiantian Liu, Jing Li, Bernhard Thomaszewski, and Ladislav Kavan. 2018b. FEPR: Fast energy projection for real-time simulation of deformable objects. *ACM Transactions on Graphics* 37(4).

Ding, Kai, Libin Liu, Michiel van de Panne, and KangKang Yin. 2015. Learning reduced-order feedback policies for motion skills. In *Proceedings of the 14th acm siggraph / eurographics symposium on computer animation*, 83–92. SCA '15, New York, NY, USA: Association for Computing Machinery.

Doersch, Carl. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

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

Ekman, Paul, and Wallace V Friesen. 1978. Facial action coding system. *Environmental Psychology & Nonverbal Behavior*.

Fan, Ye, Joshua Litven, and Dinesh K Pai. 2014. Active volumetric musculoskeletal systems. *Proc. of ACM SIGGRAPH* 33(4):152.

Faure, François, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, et al. 2012. Sofa: A multi-model framework for interactive physical simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*, 283–321. Springer.

Flynn, Cormac, Ian Stavness, John Lloyd, and Sidney Fels. 2015. A finite element model of the face including an orthotropic skin model under in vivo tension. *Computer methods in biomechanics and biomedical engineering* 18(6):571–582.

Flynn, Cormac, Andrew J Taberner, Poul MF Nielsen, and Sidney Fels. 2013. Simulating the three-dimensional deformation of in vivo facial skin. *Journal of the mechanical behavior of biomedical materials* 28:484–494.

Fratarcangeli, Marco, Valentina Tibaldo, and Fabio Pellacini. 2016. Vivace: A practical gauss-seidel method for stable soft body dynamics. *ACM Transactions on Graphics* 35(6):1–9.

Geilinger, Moritz, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. Add: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)* 39(6):1–15.

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

Gladilin, Evgeny. 2003. Biomechanical modeling of soft tissue and facial expressions for craniofacial surgery planning. *Freien University, Berlin*.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Hahn, David, Pol Banzet, James M Bern, and Stelian Coros. 2019. Real2sim: Viscoelastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG)* 38(6):1–13.

Hu, John, Chu-Yin Chang, Neil Tardella, Janey Pratt, James English, et al. 2006. Effectiveness of haptic feedback in open surgery simulation and training systems. *Stud. Health Technol. Inform* 119:213–218.

Hu, Yuanming, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. Difftaichi: Differentiable programming for physical simulation. In *International conference on learning representations*.

Hu, Yuanming, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. 2019. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 international conference on robotics and automation (icra)*, 6265–6271. IEEE.

Ichim, Alexandru-Eugen, Petr Kadleček, Ladislav Kavan, and Mark Pauly. 2017. Phace: Physics-based face modeling and animation. *ACM Transactions on Graphics (TOG)* 36(4):153.

Ichim, Alexandru-Eugen, Ladislav Kavan, Merlin Nimier-David, and Mark Pauly. 2016. Building and animating user-specific volumetric face rigs. In *Proceedings of the acm siggraph/eurographics symposium on computer animation*, 107–117. SCA '16, Goslar, DEU: Eurographics Association.

Irving, Geoffrey, Joseph Teran, and Ronald Fedkiw. 2004. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 acm siggraph/eurographics symposium on computer animation*, 131–140.

Jaberi, Mehrad, Jad Abi-Rafeh, Yehuda Chocron, Dino Zammit, Becher Al-Halabi, and Mirko S Gilardino. 2021. Smart assessment tool: an innovative approach for objective assessment of flap designs. *Plastic and Reconstructive Surgery* 148(5): 837e–840e.

Joodaki, Hamed, and Matthew B Panzer. 2018. Skin mechanical properties and modeling: A review. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine* 232(4):323–343.

Kadlecek, Petr, and Ladislav Kavan. 2019. Building accurate physics-based face models from data. In *Symposium on computer animation*.

Kazan, Roy, Shantale Cyr, Thomas M Hemmerling, Samuel J Lin, and Mirko S Gilardino. 2017. The evolution of surgical simulation: the current state and future avenues for plastic surgery education. *Plastic and reconstructive surgery* 139(2): 533e–543e.

Kee, Min Hyung, Kiwon Um, Wooseok Jeong, and Junghyun Han. 2021. Constrained projective dynamics. *ACM Transactions on Graphics* 40(4):1–12.

Kingma, Diederik P., and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd international conference on learning representations, ICLR 2014, banff, ab, canada, april 14-16, 2014, conference track proceedings*.

Kite, Amy C, Morgan Yacoe, and Jennifer L Rhodes. 2018. The use of a novel local flap trainer in plastic surgery education. *Plastic and Reconstructive Surgery Global Open* 6(6).

Klár, Gergely, Andrew Moffat, Ken Museth, and Eftychios Sifakis. 2020. Shape targeting: A versatile active elasticity constitutive model. In *Special interest group on computer graphics and interactive techniques conference talks*. SIGGRAPH '20, Association for Computing Machinery.

Komaritzan, Martin, and Mario Botsch. 2018. Projective skinning. *Proc. ACM Comput. Graph. Interact. Tech.* 1(1).

———. 2019. Fast projective skinning. In *Proceedings of the 12th acm siggraph conference on motion, interaction and games*. MIG '19, New York, NY, USA: Association for Computing Machinery.

Kozlov, Yeara, Derek Bradley, Moritz Bächer, Bernhard Thomaszewski, Thabo Beeler, and Markus Gross. 2017. Enriching facial blendshape rigs with physical simulation. *Comput. Graph. Forum* 36(2):75–84.

Kwan, Zhenli, Nur Nadirah Khairu Najhan, Yat Huang Yau, Yan Luximon, and Fethma M Nor. 2020. Anticipating local flaps closed-form solution on 3d face models using finite element method. *International journal for numerical methods in biomedical engineering* 36(11):e3390.

Lan, Lana, Matthew Cong, and Ronald Fedkiw. 2017. Lessons from the evolution of an anatomical facial muscle model. In *Proceedings of the acm siggraph digital production symposium*, 11. ACM.

Lan, Lei, Ran Luo, Marco Fratarcangeli, Weiwei Xu, Huamin Wang, Xiaohu Guo, Junfeng Yao, and Yin Yang. 2020. Medial elastics: Efficient and collision-ready

deformation via medial axis transform. *ACM Transactions on Graphics (TOG)* 39(3): 1–17.

Lapeer, RJ, PD Gasson, and V Karri. 2010. Simulating plastic surgery: From human skin tensile tests, through hyperelastic finite element models to real-time haptics. *Progress in biophysics and molecular biology* 103(2-3):208–216.

Le, Binh Huy, and J P Lewis. 2019. Direct delta mush skinning and variants. *ACM Trans. Graph.* 38(4).

Lee, Seunghwan, Ri Yu, Jungnam Park, Mridul Aanjaneya, Eftychios Sifakis, and Jehee Lee. 2018a. Dexterous manipulation and control with volumetric muscles. *ACM Trans. Graph.* 37(4).

Lee, Taeksang, Sergey Y Turin, Casey Stowers, Arun K Gosain, and Adrian Buganza Tepole. 2021. Personalized computational models of tissue-rearrangement in the scalp predict the mechanical stress signature of rotation flaps. *The Cleft Palate-Craniofacial Journal* 58(4):438–445.

Lee, Taeksang, Elbert E Vaca, Joanna K Ledwon, Hanah Bae, Jolanta M Topczewska, Sergey Y Turin, Ellen Kuhl, Arun K Gosain, and Adrian Buganza Tepole. 2018b. Improving tissue expansion protocols through computational modeling. *Journal of the mechanical behavior of biomedical materials* 82:224–234.

Lewis, John P, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Frederic H Pighin, and Zhigang Deng. 2014. Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports)* 1(8):2.

Li, Jiaman, Zhengfei Kuang, Yajie Zhao, Mingming He, Karl Bladin, and Hao Li. 2020a. Dynamic facial asset and rig generation from a single scan. *ACM Transactions on Graphics (TOG)* 39(6):1–18.

Li, Jing, Tiantian Liu, and Ladislav Kavan. 2018. Laplacian damping for projective dynamics. In *Proceedings of the 14th workshop on virtual reality interactions and physical simulations*, 29–36. VRIPHYS '18, Goslar, DEU: Eurographics Association.

———. 2019. Fast simulation of deformable characters with articulated skeletons in projective dynamics. In *Proceedings of the 18th annual acm siggraph/eurographics symposium on computer animation*. SCA '19, New York, NY, USA: Association for Computing Machinery.

Li, Jing, Tiantian Liu, Ladislav Kavan, and Baoquan Chen. 2021. Interactive cutting and tearing in projective dynamics with progressive cholesky updates. *ACM Transactions on Graphics* 40(6):1–12.

Li, Ruilong, Karl Bladin, Yajie Zhao, Chinmay Chinara, Owen Ingraham, Pengda Xiang, Xinglei Ren, Pratusha Prasad, Bipin Kishore, Jun Xing, et al. 2020b. Learning formation of physically-based face attributes. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 3410–3419.

Liu, Haixiang, Nathan Mitchell, Mridul Aanjaneya, and Eftychios Sifakis. 2016. A scalable schur-complement fluids solver for heterogeneous compute platforms. *ACM Transactions on Graphics (TOG)* 35(6):201.

Liu, Tiantian, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics (TOG)* 36(3):23.

Lloyd, John E, Ian Stavness, and Sidney Fels. 2012. Artisynth: A fast interactive biomechanical modeling toolkit combining multibody and finite element simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*, 355–394. Springer.

Lombardi, Stephen, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep appearance models for face rendering. *ACM Transactions on Graphics (TOG)* 37(4): 68.

Lovald, Scott T, Shelby G Topp, Jorge A Ochoa, and Curtis W Gaball. 2013. Biomechanics of the monopedicle skin flap. *Otolaryngology–Head and Neck Surgery* 149(6): 858–864.

Ly, Mickaël, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. 2020. Projective dynamics with dry frictional contact. *ACM Transactions on Graphics* 39(4):57:1–57:8.

Maas, Steve A, Gerard A Ateshian, and Jeffrey A Weiss. 2017. Febio: History and advances. *Annual review of biomedical engineering* 19:279–299.

Markenscoff, X, and IV Yannas. 1979. On the stress-strain relation for skin. *Journal of biomechanics* 12(2):127–129.

Martin, Sebastian, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based elastic materials. In *Acm siggraph 2011 papers*. SIGGRAPH '11, New York, NY, USA: Association for Computing Machinery.

McAdams, Aleka, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.* 30(4).

McNamara, Antoine, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. *ACM Trans. Graph.* 23(3):449–456.

Mitchell, Nathan, Mridul Aanjaneya, Rajsekhar Setaluri, and Eftychios Sifakis. 2015a. Non-manifold level sets: A multivalued implicit surface representation with applications to self-collision processing. *ACM Transactions on Graphics (TOG)* 34(6):1–9.

Mitchell, Nathan, Court Cutting, and Eftychios Sifakis. 2015b. GRIDiron: An interactive authoring and cognitive training foundation for reconstructive plastic surgery procedures. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH)*.

Mitchell, Nathan M, Timothy W King, Aaron Oliker, Eftychios D Sifakis, et al. 2016. A real-time local flaps surgical simulator based on advances in computational algorithms for finite element models. *Plastic and reconstructive surgery* 137(2): 445e–452e.

Molino, Neil, Zhaosheng Bao, and Ron Fedkiw. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Transactions on Graphics (TOG)* 23(3):385–392.

Müller, Matthias, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2014. Strain Based Dynamics. In *Proceedings of the acm siggraph/eurographics symposium on computer animation - sca '14*, 149–157.

Müller, Matthias, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. 2002. Stable real-time deformations. In *Proceedings of the 2002 acm siggraph/eurographics symposium on computer animation*, 49–54.

Müller, Matthias, and Markus Gross. 2004. Interactive virtual materials. In *Proceedings of graphics interface 2004*, 239–246. Canadian Human-Computer Communications Society.

Müller, Matthias, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2006. Position based dynamics. *3rd Workshop in Virtual Reality Interactions and Physical Simulations, VRIPHYS 2006* 71–80.

Narain, Rahul, Matthew Overby, and George E Brown. 2016. ADMM $\supseteq$ Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proceedings of the acm siggraph/eurographics symposium on computer animation*, 21–28. Eurographics Association.

Nardinocchi, Paola, and Luciano Teresi. 2007. On the active response of soft living tissues. *Journal of Elasticity* 88(1):27–39.

Naveed, Hasan, Richard Hudson, Manaf Khatib, and Fernando Bello. 2018. Basic skin surgery interactive simulation: system description and randomised educational trial. *Advances in Simulation* 3(1):1–9.

Oliker, Aaron, and Court Cutting. 2005. The role of computer graphics in cleft lip and palate education. *Seminars in Plastic Surgery* 19(04):286–293.

Overby, Matthew, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Transactions on Visualization and Computer Graphics* 23(10):2222–2234.

Peng, Yue, Bailin Deng, Juyong Zhang, Fanyu Geng, Wenjie Qin, and Ligang Liu. 2018. Anderson acceleration for geometry optimization and physics simulation. *ACM Transactions on Graphics* 37(4). 1805.05715.

Pieper, Steven D, DR Laub, and Joseph M Rosen. 1995. A finite-element facial model for simulating plastic surgery. *Plastic and Reconstructive Surgery* 96:1100–1100.

Podolsky, Dale J, David M Fisher, Karen W Wong, Thomas Looi, James M Drake, and Christopher R Forrest. 2017. Evaluation and implementation of a high-fidelity cleft palate simulator. *Plastic and reconstructive surgery* 139(1):85e–96e.

Powell, Allison R, Sudharsan Srinivasan, Glenn Green, Jennifer Kim, and David A Zopf. 2019. Computer-aided design, 3-d–printed manufacturing, and expert validation of a high-fidelity facial flap surgical simulator. *JAMA Facial Plastic Surgery*.

Raposio, Edoardo, and Rolf EA Nordström. 1998. Biomechanical properties of scalp flaps and their correlations to reconstructive and aesthetic surgery procedures. *Skin Research and Technology* 4(2):94–98.

Ridge, MD, and V Wright. 1965. A bio-engineering study of the mechanical properties of human skin in relation to its structure. *British Journal of Dermatology* 77(12):639–649.

Roussellet, Valentin, Nadine Abu Rumman, Florian Canezin, Nicolas Mellado, Ladislav Kavan, and Loïc Barthe. 2018. Dynamic implicit muscles for character skinning. *Computers & Graphics* 77:227–239.

Rubin, MB, and SR Bodner. 2002. A three-dimensional nonlinear model for dissipative response of soft tissue. *International Journal of Solids and Structures* 39(19): 5081–5099.

Sano, Hitomi, Yu Hokazono, and Rei Ogawa. 2018. Distensibility and gross elasticity of the skin at various body sites and association with pathological scarring: a case study. *The Journal of clinical and aesthetic dermatology* 11(6):15.

Satava, MD, and M Richard. 2001. Surgical education and surgical simulation. *World journal of surgery* 25(11):1484–1489.

Schendel, Stephen, Kevin Montgomery, Andrea Sorokin, and Giancarlo Lionetti. 2005. A surgical simulator for planning and performing repair of cleft lips. *Journal of Cranio-Maxillofacial Surgery* 33(4):223–228.

Schmidt, Michael, and Hod Lipson. 2009. Distilling free-form natural laws from experimental data. *Science* 324(5923):81–85.

Schwartz, Gabriel, Shih-En Wei, Te-Li Wang, Stephen Lombardi, Tomas Simon, Jason Saragih, and Yaser Sheikh. 2020. The eyes have it: an integrated eye and face model for photorealistic facial animation. *ACM Transactions on Graphics (TOG)* 39(4):91–1.

de Sena, David P, Daniela D Fabricio, Maria Helena I Lopes, and Vinicius D da Silva. 2013. Computer-assisted teaching of skin flap surgery: validation of a mobile platform software for medical students. *PloS one* 8(7):e65833.

Sifakis, Eftychios, and Jernej Barbic. 2012. Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*. SIGGRAPH '12, New York, NY, USA: Association for Computing Machinery.

Sifakis, Eftychios, Jeffrey Hellrung, Joseph Teran, Aaron Oliker, et al. 2009. Local flaps: A real-time finite element based solution to the plastic surgery defect puzzle. In *Medicine meets virtual reality 17*, 313–318. IOS Press.

Sifakis, Eftychios, Igor Neverov, and Ronald Fedkiw. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. In *Proc. of acm siggraph*, vol. 24, 417–425.

Smith, Breannan, Chenglei Wu, He Wen, Patrick Peluse, Yaser Sheikh, Jessica K Hodgins, and Takaaki Shiratori. 2020. Constraining dense hand surface tracking with elasticity. *ACM Transactions on Graphics (TOG)* 39(6):1–14.

Soler, Carlota, Tobias Martin, and Olga Sorkine-Hornung. 2018. Cosserat Rods with Projective Dynamics. *Computer Graphics Forum* 37(8):137–147.

Spüler, Martin, Nerea Irastorza Landa, Andrea Sarasola Sanz, and Ander Ramos-Murguialday. 2016. Extracting muscle synergy patterns from emg data using autoencoders. In *Artificial neural networks and machine learning – icann 2016*, 47–54.

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

Stavness, Ian, Mohammad Ali Nazari, Cormac Flynn, Pascal Perrier, Yohan Payan, John E Lloyd, and Sidney Fels. 2014. Coupled biomechanical modeling of the face, jaw, skull, tongue, and hyoid bone. In *3d multiscale physiological human*, 253–274. Springer.

Stomakhin, Alexey, Russell Howes, Craig Schroeder, and Joseph M Teran. 2012. Energetically consistent invertible elasticity. In *Proceedings of the acm siggraph/eurographics symposium on computer animation*, 25–32. Eurographics Association.

Tan, Jie, Greg Turk, and C Karen Liu. 2012. Soft body locomotion. *ACM Transactions on Graphics (TOG)* 31(4):26.

Tang, Wen, and Tao Ruan Wan. 2014. Constraint-based soft tissue simulation for virtual surgical training. *IEEE Transactions on Biomedical Engineering* 61(11): 2698–2706.

Taylor, Steven R, and CW David Chang. 2016. Gelatin facial skin simulator for cutaneous reconstruction. *Otolaryngology–Head and Neck Surgery* 154(2):279–281.

Tena, J. Rafael, Fernando De la Torre, and Iain Matthews. 2011. Interactive region-based linear 3d face models. In *Acm siggraph 2011 papers*. SIGGRAPH '11, New York, NY, USA: Association for Computing Machinery.

Teng, Yun, Miguel A Otaduy, and Theodore Kim. 2014. Simulating articulated subspace self-contact. *ACM Transactions on Graphics (TOG)* 33(4):1–9.

Teran, J., S. Blemker, Ng V. Thow Hing, and R. Fedkiw. 2003. Finite volume methods for the simulation of skeletal muscle. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2003* M:68–75.

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

Teran, Joseph, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005b. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 acm siggraph/eurographics symposium on computer animation*, 181–190.

Teschner, M., S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. 2005. Collision detection for deformable objects. *Computer Graphics Forum* 24(1):61–81. `https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2005.00829.x`.

Then, C, B Stassen, K Depta, and G Silber. 2017. New methodology for mechanical characterization of human superficial facial tissue anisotropic behaviour in vivo. *Journal of the Mechanical Behavior of Biomedical Materials* 71:68–79.

Vaillant, Rodolphe, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. 2013. Implicit skin-

ning: real-time skin deformation with contact modeling. *ACM Transactions on Graphics (TOG)* 32(4):1–12.

Veronda, DR, and RA Westmann. 1970. Mechanical characterization of skin—finite deformations. *Journal of biomechanics* 3(1):111–124.

Wallace, Lauren, Nicholas Raison, Faisal Ghumman, Aidan Moran, Prokar Dasgupta, and Kamran Ahmed. 2017. Cognitive training: how can it be adapted for surgical education? *The Surgeon* 15(4):231–239.

Wang, Huamin. 2015. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Transactions on Graphics* 34(6):1–9.

Wang, Huamin, and Yin Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Transactions on Graphics* 35(6).

Wang, Qisi, Yutian Tao, Eric Brandt, Court Cutting, and Eftychios Sifakis. 2021. Optimized processing of localized collisions in projective dynamics. *Computer Graphics Forum* 40(6):382–393. `https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14385`.

Wang, Qisi, Yutian Tao, Court Cutting, and Eftychios Sifakis. 2022. A computer based facial flaps simulator using projective dynamics. *Computer Methods and Programs in Biomedicine* 218:106730.

Wei, Shih-En, Jason Saragih, Tomas Simon, Adam W Harley, Stephen Lombardi, Michal Perdoch, Alexander Hypes, Dawei Wang, Hernan Badino, and Yaser Sheikh. 2019. Vr facial animation via multiview image translation. *ACM Transactions on Graphics (TOG)* 38(4):1–16.

Weickenmeier, Johannes, Mahmood Jabareen, and Edoardo Mazza. 2015. Suction based mechanical characterization of superficial facial soft tissues. *Journal of biomechanics* 48(16):4279–4286.

Weiss, Jeffrey A, Bradley N Maker, and Sanjay Govindjee. 1996. Finite element implementation of incompressible, transversely isotropic hyperelasticity. *Computer methods in applied mechanics and engineering* 135(1):107–128.

Wojtan, Chris, Peter J. Mucha, and Greg Turk. 2006. Keyframe control of complex particle systems using the adjoint method. In *Sca '06: Proceedings of the 2006 acm siggraph/eurographics symposium on computer animation*, 15–23. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Wu, Chenglei, Derek Bradley, Markus Gross, and Thabo Beeler. 2016. An anatomically-constrained local deformation model for monocular face capture. *ACM transactions on graphics (TOG)* 35(4):1–12.

Wu, Jun, Rüdiger Westermann, and Christian Dick. 2015. A survey of physically based simulation of cuts in deformable bodies. *Computer Graphics Forum* 34(6): 161–187. `https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12528`.

Zajac, Felix E. 1989. Muscle and tendon properties models scaling and application to biomechanics and motor. *Critical reviews in biomedical engineering* 17(4):359–411.

Zhang, Jinao, Yongmin Zhong, and Chengfan Gu. 2017. Deformable models for surgical simulation: a survey. *IEEE reviews in biomedical engineering* 11:143–164.

Zhang, Xiaotian, Fan Kiat Chan, Tejaswin Parthasarathy, and Mattia Gazzola. 2019. Modeling and simulation of complex dynamic musculoskeletal architectures. *Nature communications* 10(1):1–12.