

AMERICAN UNIVERSITY OF BEIRUT

EFFICIENT SUTURING IN DEFORMABLE MODELS

by

GEORGES EMILE YOUNES

A thesis

submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
September 2021

AMERICAN UNIVERSITY OF BEIRUT

EFFICIENT SUTURING IN DEFORMABLE MODELS

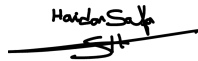
by
GEORGES EMILE YOUNES

Approved by:



Dr. George Turkiyyah, Professor
Computer Science

Advisor



Dr. Haidar Safa, Professor
Computer Science

Committee Member



Dr. Wassim El-Hajj, Associate Professor
Computer Science

Committee Member

Date of thesis defense: September 4, 2021

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

Student Name: Younes Georges Emile
Last First Middle

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of my thesis; (b) include such copies in the archives and digital repositories of the university; and (c) make freely available such copies to third parties for research or educational purposes:

- As of the date of submission of my thesis.
- One year from the date of submission of my thesis.
- Two years from the date of submission of my thesis.
- Three years from the date of submission of my thesis.

Georges September 17, 2021
Signature Date

Acknowledgments



An Abstract of the Thesis of

Georges Emile Younes

for

Master of Science

Major: Computer Science

Title: Efficient Suturing in Deformable Models

Surgical procedures are among the most complex medical interventions routinely performed. The suturing task, ubiquitous to most surgical interventions, has a steep learning curve. With the increased focus on minimally invasive interventions, the suturing task has become more complex than ever and compulsory training on virtual simulators has become the norm. However, simulating the suturing task is not trivial, as it involves the simulation of needle-tissue and thread-tissue interaction, as well as contact between sutured boundaries.

In this work, robust methods are proposed for efficiently simulating suturing using a linear elastic finite element-based approach. Important aspects of our proposed method include modeling contact conditions using complementarity relations and simulating needle driving in real-time, without introducing any new mesh elements, through the use of adjacency relationships inherent to manifold meshes and decomposing rigid body motion into two disjoint sliding and sticking components. In addition, a computationally efficient formulation of thread pulling is presented which relies on distance linearization and achieves plausible results.

These techniques are implemented and tested in a cross-platform prototype system, that allows for interactive simulation of suturing with high-resolution models through a simplified anastomosis scenario while making use of multiple human-computer interaction devices and immersive rendering techniques.

Contents

Acknowledgments	iv
Abstract	v
Contents	vi
Figures	viii
Tables	ix
Algorithms	x
Abbreviations	xi
Notation	xii
1 Introduction	1
1.1 Motivation	1
1.2 Contribution	3
1.3 Outline	4
2 Background	6
2.1 Mesh Representation	6
2.1.1 Simplicies and Complexes	6
2.1.2 Indexing of Simplicies	9
2.1.3 Multi-Level Queries	12
2.2 Deformable Models	16
2.2.1 Linear Elasticity	17
2.2.2 Finite Element Method	19
2.2.3 Corotational Method	22
2.2.4 Algebraic Constraints	23
2.2.5 Sparse Solvers	25
2.2.6 Collision Response	27
2.3 Related Work	30
3 Methods	34
3.1 Motion Decomposition	34
3.1.1 Needle Parametrization	37
3.2 Needle Driving	40
3.2.1 Incremental Tracking	41
3.2.1.1 Tip Tracking	42
3.2.1.2 Intersection Tests	45
3.2.2 History Tracking	46
3.2.2.1 Special Cases	50

3.2.3	Boundary Puncture	51
3.2.3.1	Motion Playback	52
3.2.4	Transverse Deformation	53
3.2.4.1	Slipping and Friction	55
3.3	Thread Pulling	56
3.3.1	Pull Modeling	58
3.3.2	Distance Linearization	60
4	Results	63
5	Conclusion	70
5.1	Summary	70
5.2	Future Work	71
	References	73

Figures

2.1	Compact representation of a 3-simplex	10
2.2	Opposite facet adjacency relation	13
2.3	Link and open star of faces of a vertex and an edge	14
2.4	Discretization of a continuous domain into finite elements	20
2.5	Vertex–Face and Edge–Edge intersections	28
2.6	Deformation of two soft slabs with contact handling constraints	29
3.1	Linear and semi-circular motion decomposition	37
3.2	Tracking a straight needle inside a soft tissue mesh	41
3.3	Tip tracking through a tetrahedral slab	44
3.4	Four cases for addition, removal, and update of history cells	47
3.5	Three possible inter-cell crossings	50
3.6	Tip tracking through a tetrahedral slab with edge intersections	51
3.7	Needle driving through two slabs	54
3.8	Sequence of puncture points in a running suture	58
3.9	Distance linearization for a thread segment	61
3.10	Thread pulling for tightening a suture	62
4.1	Overall system architecture	63
4.2	Needle motion relay from end-user input devices	64
4.3	Stereo pair and red-cyan anaglyph using off-axis projection	64
4.4	Bi-manual frame by Mimic Technologies	65
4.5	Needle driving using a 1/2-circle round body needle	67
4.6	Closing of two boundaries using 20 puncture points	68
4.7	Closing of two boundaries using 50 puncture points	69

Tables

2.1	Indexing and orientation of the half-facets in a cell	11
2.2	Indexing and orientation of the half-edges in a cell	11

Algorithms

2.1	Star of facets for a vertex query	15
2.2	Open star of facets for a vertex query	15
2.3	Link of faces for a vertex query	15
2.4	Open star of facets for an edge query	15
2.5	Link of facets for an edge query	16
3.1	Tip tracking through a tetrahedral mesh	43
3.2	Simplified tip tracking through a tetrahedral mesh	44
3.3	History tracking with forward sliding	48
3.4	History tracking with forward/backward sliding	49
3.5	Different steps of needle driving	57

Abbreviations

AR	augmented reality	3
BVH	bounding volume hierarchy	27
CHE	compact half edge	12
CHF	compact half face	10, 12, 14
DOF	degrees of freedom	64
EE	edge-edge	27–30
FEM	finite element method	19, 20, 22, 30–32
FPS	frames per second	66
GPU	graphics processing unit	65
I/O	input/output	65
MR	mixed reality	3
MVC	model-view-controller	63
PD	polar decomposition	23
SOT	sorted opposite table	14
SVD	singular value decomposition	22, 23
VF	vertex-face	27–29, 51, 53
VR	virtual reality	2, 3
XR	extended reality	3

Notation

A	matrix
a	vector
<i>a</i>	scalar
\hat{u}	unit vector
I	identity matrix
0	zero matrix
A	container
\square^T	transpose of \square
$\square^{[e]}$	value of \square for element e
$\square^{[k]}$	value of \square at step k
$\square^{[e,k]}$	value of \square for element e at step k

to Charles, in memoriam

Chapter 1

Introduction

“The good news is that virtual reality is here. The bad news is that something is still missing.”

—Mychilo Cline

Suturing is a fundamental operation in surgery and is an ever-present task in procedures ranging from closing lacerations and stitching wounds in open surgery, to anastomosis and closing incisions after tissue removal in laparoscopic and robot-assisted procedures. The techniques for planning suturing tasks and the manual dexterity and hand-eye coordination involved in executing them are essential skills that student surgeons and practitioners develop through hours upon hours of training and practice.

1.1 Motivation

In the context of laparoscopic and robot-assisted surgery, with their constrained workspace due to instrument kinematics, this practice often has a steep learning curve, and it takes a substantial amount of training to perfect the necessary skills. The practice generally takes place on suturing boards, phantom models, or on live animals, but there has been significant interest in the use of simulation-based training tools due to the flexibility in the training scenarios that the simulators can provide, their convenience, and their ultimate cost-effectiveness. A number of initiatives have been formulated by official bodies, such as the American College of Surgeons, to promote simulation-based training, including virtual reality-based simulation [35].

A major advantage of virtual simulation when compared to classical techniques is that it provides data and parameters which can be used to design new

metrics, allowing for objective assessment of surgical skills. The literature on the objective assessment of surgical skills is rich in attempting to find the optimal approach for training surgeons on particular tasks. In addition, virtual simulators are flexible, accessible, and cost-effective when it comes to creating new scenarios and procedures with increasing degrees of difficulties and have proven to transfer the skills required for laparoscopic and robot-assisted surgery as demonstrated in [51, 54, 55, 56, 57, 63, 64] and can even outperform traditional box trainers in some instances [46, 60]. The current pandemic is another major reason for investing in surgical simulators as they are one of the few available training tools in socially distanced societies.

Despite major efforts, there is not yet a satisfactory and openly available suturing simulation model, that has the sufficient realism and interactive response rate, suitable for training medical students and related professionals and that can run on commodity hardware such as integrated head-mounted virtual reality (VR) headsets or limited environments such as web browsers. The difficulties of modeling suturing operations come from a number of factors. An inextensible thread, essentially massless, with almost negligible bending stiffness is driven by a rigid tool in a highly deformable anatomically complex, heterogeneous, anisotropic soft tissue environment. The geometry of the environment is continuously changing because of the needle movement and the general contact conditions between the deforming tissue being sutured as well as between tools, threads, and tissue. These simulations have to be performed on a limited computational budget to allow for interactive use operations that support real-time visual feedback and haptic interaction.

While the incorporation of mass, bending stiffness, and axial flexibility in the needle and thread make them more amenable to more general and standard mechanics-based finite element simulations, the additional computational efforts needed in the solution are prohibitive for real-time simulation and the added realism is not central to this work. In addition, cutting, fracture, and tearing effects are

ignored to avoid the computationally intense task of mesh subdivision (re-meshing). The scope is limited to a quasi-static linear elastic tissue model, a rigid needle model, and a linear geometric thread model, to be able to highlight and test the proposed motion decomposition technique and incremental tracking procedures.

1.2 Contribution

The main goal of this work is to develop tools and techniques that augment the ever expanding body of work on extended reality (XR)—a term that covers VR, augmented reality (AR), and mixed reality (MR)—simulators for surgical procedure training and virtual suturing, in particular by providing effective and integrated models and methods for needle driving and suture simulation. The models and methods strike a good balance between high-fidelity and real-time simulations and can run at interactive rates with a low computational budget to be used in applications targeting VR headsets and web browsers.

The main contribution of this work is a computationally-effective model for realistic suture simulation. More specifically, given neighboring soft tissue volumes with a gap between them, a thread can be inserted into the volumes to bring them together at two points, thus closing the opening in real-time while the soft tissue follows a mechanically plausible behavior. The proposed approach introduces efficient geometric operations to allow a needle and a thread to traverse multiple boundaries and builds the appropriate constraints to use in a quasi-static linear finite element model.

The rigid nature of the needle is used to decompose driving motion into sliding and deforming (sticking) components, and handle them separately. A new thread pulling constraint model using distance linearization is introduced and demonstrated on a realistic example of an anastomosis procedure involving a fine mesh with multiple puncture points. The aforementioned techniques are

implemented and tested in a cross-platform prototype system that allows for interactive simulation of suturing with high-resolution models through a simplified anastomosis scenario while making use of multiple human-computer interaction devices and immersive rendering techniques.

1.3 Outline

The rest of this thesis is organized as follows: In §2, the background details of the various components required for modeling needle-tissue, tissue-tissue, and thread-tissue interactions and an overview of the literature for surgical simulation are presented. §2.1 outlines the compact face data structure used for representing soft models using tetrahedral meshes, its indexing schemes and implementation levels, and the adjacency queries essential to the incremental needle tracking technique. §2.2 uses a continuum mechanics formulation to simulate soft models using a quasi-static linear finite element method and a direct sparse solver to solve the underlying system of equations and satisfy the constraints at every step. Three-dimensional inter-boundary contacts are detected using continuous collision detection and modeled using linear complementarity constraints. §2.3 gives an overview of suture simulation and related work.

In §3, we present a motion decomposition technique and treatment of needle-tissue interaction through two separate sliding and sticking phases and presents a thread pulling model that relies on distance linearization. §3.1, explains our strategy for motion decomposition and handling each component separately for a straight and a semi-circular needle. §3.2 outlines the techniques used for tracking a parametric needle representation through a tetrahedral mesh, as it deforms soft tissue while puncturing through multiple tissue boundaries. §3.3 presents a thread pulling model, suitable for single stitches as well as running sutures, to close openings between two boundaries.

§4 gives an overview of the implementation and shows results from a prototype of the system with update rates that allow real-time performance to be achieved. Discussion of current and future work conclude the thesis in §5.

Chapter 2

Background

2.1 Mesh Representation

Efficient and compact data structures are indispensable when it comes to encoding surface and volume meshes. These data structures are usually equipped with various operators and containers to facilitate common operations such as rendering, texturing, smoothening, etc., and to attach various attributes to nodes such as color, mass, material property, etc. Unstructured oriented 2-manifold surface and 3-manifold volume meshes are used throughout this work as they provide a unified representation for surface rendering, collision detection, and physically-based simulation, and their unstructured nature can easily adapt to geometric irregularities when meshing the simulation domain. They are a realization of a special class of simplicial complexes.

2.1.1 Simplices and Complexes

A k -simplex σ is the closed convex hull of $k + 1$ linearly independent vertices, $\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_k$ (including the interior). A 0-simplex is a point (vertex), a 1-simplex is a line segment (edge), a 2-simplex is a triangle (facet), and a 3-simplex is a tetrahedron (cell) in Euclidean space. A face τ of σ is an r -simplex generated by a subset of its vertices with $r \leq k$; τ is a proper face when $r < k$, e.g. the proper faces of a tetrahedron are its vertices, edges, and facets. A simplex is incident to its faces and the faces bound the simplex; the incidence relation is denoted by $\tau \preceq \sigma$.

Each vertex \mathbf{v}_i has a face opposite generated by $\{\mathbf{v}_j \mid i \neq j\}$ and this fact is used to parametrize the space bound by σ such that every point \mathbf{p} bound by σ is

represented through a linear combination of the $k + 1$ vertices as

$$\mathbf{p} = \sum_{i=0}^k \xi_i \mathbf{v}_i, \quad (2.1)$$

with $\xi_i \geq 0$ and $\sum_{i=0}^k \xi_i = 1$. This can be expressed concisely as

$$\begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_0 & \mathbf{v}_1 & \cdots & \mathbf{v}_k \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \xi_0 \\ \xi_1 \\ \vdots \\ \xi_k \end{bmatrix}. \quad (2.2)$$

The conditions on the values of the coordinates ensure that the generated points are contained within the simplex (convex conditions) when a proper ordering (orientation) of the vertices is used. The coordinates can be computed by solving the linear system of equations in (2.2) and are expressed explicitly using Cramer's rule where each barycentric coordinate ξ_i is a ratio between the volume of the k -simplex formed with $\mathbf{p} \cup \{\mathbf{v}_j \mid i \neq j\}$ and the volume of σ itself

$$\xi_i = \frac{\begin{vmatrix} \mathbf{v}_0 & \cdots & \mathbf{v}_{i-1} & \mathbf{p} & \mathbf{v}_{i+1} & \cdots & \mathbf{v}_k \\ 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \end{vmatrix}}{\begin{vmatrix} \mathbf{v}_0 & \mathbf{v}_1 & \cdots & \mathbf{v}_k \\ 1 & 1 & \cdots & 1 \end{vmatrix}} \quad (2.3)$$

This natural representation is essential to collision detection routines, finite element approximation techniques, and geometric coupling methods that are employed throughout this work. It mainly serves as a mean of interpolating the values assigned at the nodes such as the geometric position, texture coordinate, mass, stress, and force. The values of the coordinates can also be used to distinguish between a facet (one coordinate is zero), an edge (two coordinates are zero), and a vertex (three coordinates are zero) that are part of the boundary; interior points have four non-zero coordinates.

A k -simplicial complex Σ is a collection of r -simplices ($r \leq k$), together with all their faces, where any two simplices in Σ either intersect at a face common to both of them or do not intersect at all:

$$\forall \sigma \in \Sigma : \tau \preceq \sigma \Rightarrow \tau \in \Sigma \quad (2.4)$$

$$\forall \sigma, \tau \in \Sigma : \sigma \cap \tau = \emptyset \vee \sigma \cap \tau \preceq \sigma \wedge \sigma \cap \tau \preceq \tau \quad (2.5)$$

This ensures that the simplices are connected only at their boundaries (through a vertex, edge, or facet), and that their interiors do not intersect, *i.e.* the complex is a proper partitioning of the space that it is bounding. The star (co-boundary) of a face τ in Σ is the sub-complex consisting of all faces containing τ along with all their faces (referred to as co-faces), and the link of a face is the faces in its star that are not intersecting with it

$$\text{st}[\tau] = \{s \in \Sigma \mid \exists \sigma \in \Sigma : \tau \preceq \sigma \wedge s \preceq \sigma\}, \quad (2.6)$$

$$\text{lk}[\tau] = \{\sigma \in \Sigma \mid \sigma \in \text{st}[\tau] \wedge \tau \cap \sigma = \emptyset\}. \quad (2.7)$$

A complex is pure (homogenous) if every face in Σ is a face of a k -simplex in Σ , *i.e.* Σ is a collection of k -simplices that are glued together at common faces. A face of a pure complex is a boundary face when it belongs to a single simplex and the boundary of a complex is a sub-complex consisting of all such boundary faces; a pure simplex has a well defined boundary and as such provides a good approximation to surfaces and solids. However, pure complexes can still exhibit vertex and edge singularities which can be problematic for certain applications. These singularities can be eliminated by enforcing that every facet ($(k-1)$ -simplex) bounds either one or two k -simplices and the local neighborhood of a face can be morphed (homeomorphic) into a topological sphere or ball.

A $(k-1)$ -simplex s in Σ is manifold if and only if there are at most two k -simplices incident at s . A simplicial complex is a combinatorial k -manifold if and only if every $(k-1)$ -simplex is manifold and the link of every vertex is

combinatorially equivalent to a $(k - 1)$ -sphere or $(k - 1)$ -ball [2]. The boundary of a k -manifold is a $(k - 1)$ -manifold without boundary (watertight). A combinatorial k -manifold is orientable when a coherent orientation for all of its simplices can be assigned such that an opposite orientation is induced on the common $(k - 1)$ -face of any two adjacent k -faces.

A simplicial complex is usually equipped with adjacency queries for associating its faces to each other. The adjacency queries for a p -simplex s are the retrieval functions $r_{pq}[[s]]$ that associate a p -simplex σ with the q -simplices τ that bound common simplices [9] and are essential for any routine that has to deal with the local neighborhood of a simplex or global properties defined across the complex.

$r_{pq}[[s]]$ consists of the set of q -simplices in $\text{st}[[s]]$ when $p < q$; the set of q -simplices that are faces of s , when $p > q$; and the set of q -simplices that are faces of a common simplex of which s is a face, when $p = q$. For example, $r_{10}[[s]]$, $r_{20}[[s]]$, and $r_{30}[[s]]$ are the vertices of an edge, facet, and cell, s , respectively, and $r_{33}[[s]]$ are the cells adjacent to s through a facet. The realization of these queries dictates that the vertices and cells are indexed using a scheme that is coherent with the orientation of the faces across the complex.

We exclusively use oriented 2- and 3-manifold meshes throughout this work as they can be realized in three dimensions without self-intersections and as such are a good representation for most physical objects.

2.1.2 Indexing of Simplices

Pure complexes, as they consist of a homogenous set of simplices having the same dimension, can be represented using two containers: a geometry container, \mathbb{G} , that encodes the positions of the vertices and a topology container, \mathbb{T} , that encodes the simplices using the indices of their vertices in \mathbb{G} . Let n_k denote the number of unique k -simplices in a simplicial complex. For a tetrahedral mesh with n_3 cells and n_0 vertices, \mathbb{G} and \mathbb{T} have $3n_0$ and $4n_3$ entries, respectively; each vertex

is assigned a unique integer between 0 and $n_0 - 1$ and each tetrahedron is assigned a unique integer between 0 and $n_3 - 1$.

Compact half face (CHF) [23], an extension of the corner table data structure for triangular meshes [6], is a compact mesh representation that builds on top of these two basic containers and provides an efficient and multi-level implementation of the adjacency queries for an oriented 3-manifold as defined in §2.1.1, through the use of half-faces. A half-face is a specific orientation of a face in another face that is incident to it. A face can be indexed by multiple half-faces, e.g. an internal facet in an oriented 3-manifold tetrahedral mesh has two half-facets and a boundary edge has two half-edges with opposite orientation.

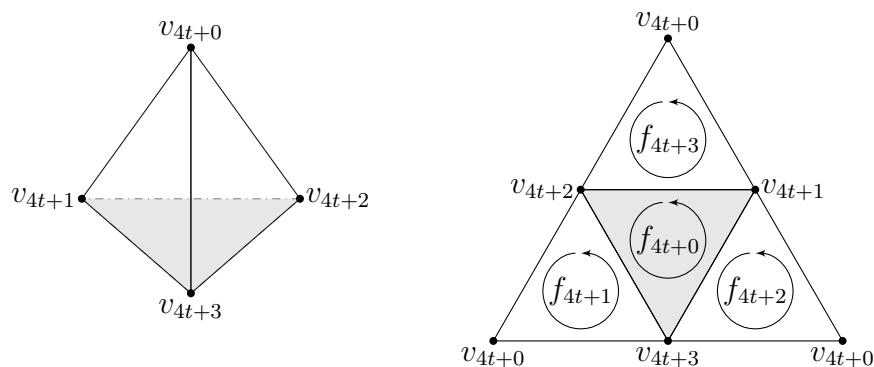


Figure 2.1 Compact representation of a 3-simplex (tetrahedron) in CHF. The orientations of the four faces are shown and the sub-indices are used to match a triangular facet with its apex (opposite vertex) (v_{4t+i} with f_{4t+i}).

CHF provides a consistent way to index all the faces and half-faces in an oriented 3-manifold mesh. Each cell is indexed using its position t in \mathbb{T} and each vertex is indexed using its position v in \mathbb{G} . An edge is indexed using its vertices (v_0, v_1) such that $v_0 < v_1$ and a facet is indexed using its vertices (v_0, v_1, v_2) such that $v_0 < v_1 < v_2$. A 3-simplex cell has 4 vertices and half-vertices, 6 edges and 12 half-edges, and 4 facets and half-facets, which results in a total of $4n_3$ half-vertices, $12n_3$ half-edges, and $4n_3$ half-facets. The half-vertices of tetrahedron t are indexed using $4t + i$ with $0 \leq i \leq 3$.

The vertices and facets of a cell t can be associated using the face opposite simplex relationship: each vertex $4t + i$ in t is associated with its opposite face and the same index is used to reference them, resulting in a compact half-face representation. With this vertex-face association, each facet can be assigned a unique index using the location of its opposite vertex in \mathbb{T} . This association is depicted in (2.1) where a cell t is incident to the vertices v_{4t+0} , v_{4t+1} , v_{4t+2} , and v_{4t+3} , and facets f_{4t+0} , f_{4t+1} , f_{4t+2} , and f_{4t+3} . The vertices of the half-facets in a cell t can be enumerated using the ordering in (2.1) with a counter-clockwise orientation of the vertices, i.e. the facets normal vectors point away from the interior of the cell.

Table 2.1 Indexing and orientation of the half-faces in a cell t .

f	v_0	v_1	v_2
$4t + 0$	$4t + 1$	$4t + 2$	$4t + 3$
$4t + 1$	$4t + 2$	$4t + 0$	$4t + 3$
$4t + 2$	$4t + 3$	$4t + 0$	$4t + 1$
$4t + 3$	$4t + 0$	$4t + 2$	$4t + 1$

A half-edge $e = (v_i, v_j)$ in a cell t is indexed using a pair of half-facet indices, (f_i, f_j) , where f_i is the half-facet index incident to e and f_j is the index of the half-facet opposite to v_i in t . The vertices of the half-edges in a cell t can be enumerated using the ordering in (2.2) using an orientation that is consistent with the half-facet orientation of (2.1).

Table 2.2 Indexing and orientation of the half-edges in a cell t .

f	e_0	e_1	e_2
$4t + 0$	$(4t + 0, 4t + 1)$	$(4t + 0, 4t + 2)$	$(4t + 0, 4t + 3)$
$4t + 1$	$(4t + 1, 4t + 2)$	$(4t + 1, 4t + 0)$	$(4t + 1, 4t + 3)$
$4t + 2$	$(4t + 2, 4t + 3)$	$(4t + 2, 4t + 0)$	$(4t + 2, 4t + 1)$
$4t + 3$	$(4t + 3, 4t + 0)$	$(4t + 3, 4t + 2)$	$(4t + 3, 4t + 1)$

Every half-vertex, half-edge, and half-facet in a mesh can be uniquely referenced using the aforementioned indexing schemes and these schemes will be

used to implement the various queries and containers required to visualize, process, tag, and assign attributes to specific simplicies in a tetrahedral mesh. The geometric coordinates of a half-simplex s with vertices v_0, v_1, \dots, v_k can be retrieved using

$$g[s] = \left[\mathbb{G}[v_0], \mathbb{G}[v_1], \dots, \mathbb{G}[v_k] \right]. \quad (2.8)$$

2.1.3 Multi-Level Queries

CHF uses four levels to implement its functionality, consuming more memory from one level to another but reducing the overall query processing time. Level 0 represents the tetrahedral soup using two containers: \mathbb{G} and \mathbb{T} . Level 1 encodes the facet-opposite relationship and augments Level 0 by an adjacency container, \mathbb{O} with $4n_3$ entries, that links opposite half-facets together; a boundary facet is linked to itself or to the bottom facet \perp depending on the application. Level 2 represents the simplicies explicitly and augments Level 1 with three additional containers: a facet container, \mathbb{F} with n_2 entries, that maps a facet to one of its half-facets; an edge container, \mathbb{E} with n_1 entries, that maps an edge to one of its half-edges (a boundary edge is mapped to a boundary half-edge); and a vertex-half-facet container, \mathbb{I} with n_0 entries, that maps a vertex to one of its incident half-facets (a boundary vertex is mapped to a boundary half-facet). Level 3 represents the boundary using a compact half edge (CHE) representation [24] which follows similar principals for representing a 2-manifold mesh, *i.e.* a 2-simplex (triangle) is compactly represented as three vertices and every edge is associated with its opposite vertex.

The half-facet opposite relation encoded using \mathbb{O} at Level 1 is used in accelerating the different queries that CHF provides, namely $r_{33}[[t]]$, which returns for a cell t its adjacent cells, can be realized in constant time without having to iterate over all the cells to find the adjacent ones; \mathbb{O} can be used to directly pick-up these cells. \mathbb{O} is also used to extract the boundary facets. The explicit representation of simplicies facilitated by Level 2 allow us to assign mesh-level

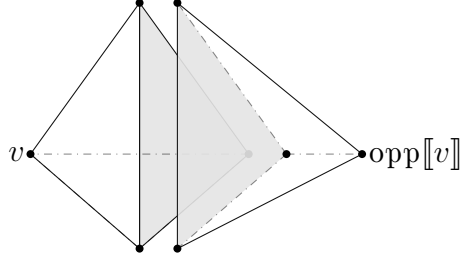


Figure 2.2 Opposite facet adjacency relation. This relation associates the two apexes of a internal facet together. Each apex corresponds to an opposite half-facet.

properties shared by all half-simplicies, such as the normal at a facet, the texture coordinates at a vertex, and the material property at a cell. The explicit representation of the boundary at Level 3 is used to extract the boundary facets and vertices for surface rendering and collision detection routines.

Each half-facet f in a cell has a next, previous, and middle half-facet in that cell, and for an interior half-facets an opposite half-facet in a an adjacent cell:

$$\text{next}[[f]] = 4(f \text{ div } 4) + (f + 1) \text{ mod } 4 \quad (2.9)$$

$$\text{mid}[[f]] = 4(f \text{ div } 4) + (f + 2) \text{ mod } 4 \quad (2.10)$$

$$\text{prv}[[f]] = 4(f \text{ div } 4) + (f + 3) \text{ mod } 4 \quad (2.11)$$

$$\text{opp}[[f]] = \mathbb{O}[[f]] \quad (2.12)$$

The next, previous, and middle half-facets all share one of their edges with the original half-facet.

Each half-edge e in a half-facet f and cell t has a next and previous half-edge in that half-facet, a mate half-edge in that cell, and for an interior half-edge a radial half-edge in an adjacent cell:

$$\text{next}[[e]] = (e_0, 4(e_0 \text{ div } 4) + \mathbf{P}[e_1 \text{ mod } 4, e_0 \text{ mod } 4]), \quad (2.13)$$

$$\text{prv}[[e]] = (e_0, 4(e_0 \text{ div } 4) + \mathbf{P}[e_0 \text{ mod } 4, e_1 \text{ mod } 4]), \quad (2.14)$$

$$\text{mte}[[e]] = (\text{prv}[[e]]_1, \text{next}[[e]]_1) \quad (2.15)$$

$$\text{rad}[[e]] = (\text{opp}[[e_0]], \text{next}[(\text{opp}[[e_0]], e_1)]_1), \quad (2.16)$$

where \mathbf{P} represents the relative orientations of the half-edges inside a cell:

$$\mathbf{P} = \begin{bmatrix} - & 3 & 1 & 2 \\ 2 & - & 3 & 0 \\ 3 & 0 & - & 1 \\ 1 & 2 & 0 & - \end{bmatrix}. \quad (2.17)$$

The repeated application of the mate and radial relationships will iterate over all the half-edges of a given edge. Note that a boundary half-edge does not have a radial half-edge and the result of the radial query is the sentinel value (\perp, \perp) .

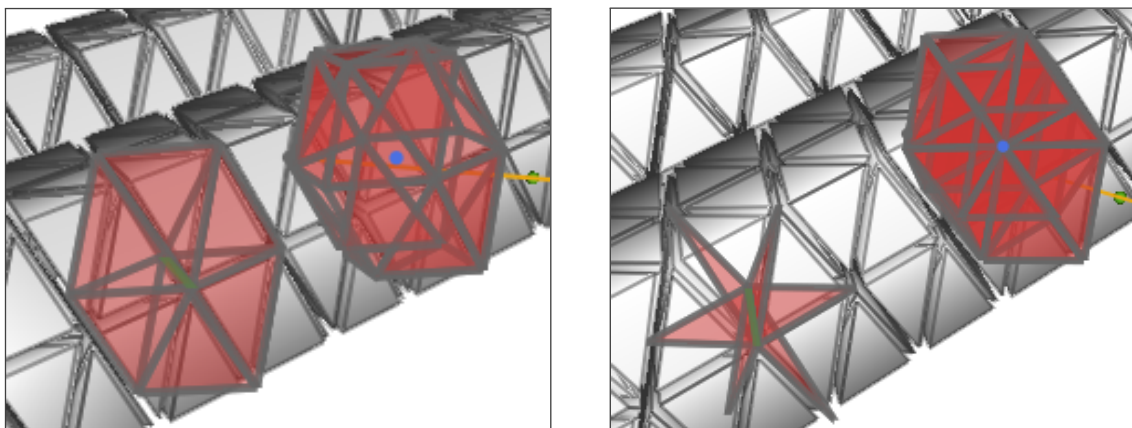


Figure 2.3 Link (*left*) and open star (*right*) of faces for a vertex (*blue*) and an edge (*green*), respectively, in a tetrahedral mesh.

CHF is a lazy data structure: the entire memory required can be initially reserved, then incrementally populated with every query and every level. The queries can be precomputed or cached using memoization to reduce their runtime and the whole memory representation can be serialized and loaded with the initial scene. Further space savings are possible by compressing the binary representation of vertex indices using an sorted opposite table (SOT) format as described in [42].

We extend CHF with two additional queries that were not addressed in the original proposal: link of facets for a vertex, \bar{r}_{02} outlined in {2.3} and link of facets for an edge, \bar{r}_{12} outlined in {2.5}. In {2.3}, the link and open star of facets are visualized for a vertex and an edge.

Algorithm 2.1 r_{02} query (star of faces for a vertex).

```
1: function  $r_{02}(v)$ 
2:    $\mathbb{S} \leftarrow \{\}$ 
3:   for  $t$  in  $r_{03}[[v]]$  do
4:     for  $i = 0$  to  $3$  do
5:        $\mathbb{S} \leftarrow \mathbb{S} \cup 4t + i$ 
6:   return  $\mathbb{S}$ 
```

Algorithm 2.2 \underline{r}_{02} query (open star of facets for a vertex).

```
1: function  $\underline{r}_{02}(v)$ 
2:    $\mathbb{S} \leftarrow \{\}$ 
3:   for  $t$  in  $r_{03}[[v]]$  do
4:     for  $i = 0$  to  $3$  do
5:       if  $r_{30}[[t, i]] \neq v$  then
6:          $\mathbb{S} \leftarrow \mathbb{S} \cup 4t + i$ 
7:   return  $\mathbb{S}$ 
```

Algorithm 2.3 \bar{r}_{02} query (link of faces for a vertex).

```
1: function  $\bar{r}_{02}(v)$ 
2:    $\mathbb{S} \leftarrow \{\}$ 
3:   for  $t$  in  $r_{03}[[v]]$  do
4:     for  $i = 0$  to  $3$  do
5:       if  $r_{30}[[t, i]] = v$  then  $[[v \text{ is opposite to facet } 4t + i]]$ 
6:          $\mathbb{S} \leftarrow \mathbb{S} \cup 4t + i$ 
7:   return  $\mathbb{S}$ 
```

Algorithm 2.4 \underline{r}_{12} query (open star of facets for an edge).

```
1: function  $\underline{r}_{12}(e)$ 
2:    $\mathbb{S} \leftarrow \{\}$ 
3:    $a \leftarrow e$ 
4:    $\text{EdgeSweep}(\mathbb{S}, a, e)$ 
5:   if  $a_0 = \perp$  then  $[[e \text{ is boundary; reverse direction}]]$ 
6:      $a \leftarrow \text{rad}[[e]]$ 
7:      $\text{EdgeSweep}(\mathbb{S}, a, e)$ 
8:   return  $\mathbb{S}$ 
```

```
1: procedure  $\text{EdgeSweep}(\mathbb{S}, a, e)$ 
2:   repeat
3:      $\mathbb{S} \leftarrow \mathbb{S} \cup a_0$ 
4:      $a \leftarrow \text{mte}[[a]]$ 
5:      $\mathbb{S} \leftarrow \mathbb{S} \cup a_0$ 
6:      $a \leftarrow \text{rad}[[a]]$ 
7:   until  $a_0 = \perp$  or  $a_0 = e_0$  do
```

Algorithm 2.5 \bar{r}_{12} query (link of facets for an edge).

```

1: function  $\bar{r}_{12}(e)$ 
2:    $\mathbb{S} \leftarrow \{\}$ 
3:    $a \leftarrow e$ 
4:   EdgeSweep( $\mathbb{S}, a, e$ )
5:   if  $a_0 = \perp$  then                                      $\llbracket e \text{ is boundary; reverse direction} \rrbracket$ 
6:      $a \leftarrow \text{rad}[[e]]$ 
7:     EdgeSweep( $\mathbb{S}, a, e$ )
8:   return  $\mathbb{S}$ 

```

```

1: procedure EdgeSweep( $\mathbb{S}, a, e$ )
2:   repeat
3:      $\mathbb{S} \leftarrow \mathbb{S} \cup a_1$ 
4:      $a \leftarrow \text{mte}[[a]]$ 
5:      $\mathbb{S} \leftarrow \mathbb{S} \cup a_1$ 
6:      $a \leftarrow \text{rad}[[a]]$ 
7:   until  $a_0 = \perp$  or  $a_0 = e_0$  do

```

2.2 Deformable Models

A soft body can undergo deformations and is contrasted with a rigid body that has a fixed geometry. Soft bodies are associated with deformable models that are represented by a finite set of geometric nodes and associated rules that governs their deformations. Physically-based deformable models are used to model tissue behavior through the laws of continuum mechanics for elastic solids.

The displacements, $\mathbf{u}^{[k]}$, encoding the deformations for a soft body with nodes \mathbf{x} are computed at each step with respect to the initial configuration (rest state), $\mathbf{x}^{[0]}$ such that:

$$\mathbf{x}^{[k]} = \mathbf{x}^{[0]} + \mathbf{u}^{[k]}. \quad (2.18)$$

Elastic models are a specific category of physically-based models that, after being deformed, return to their original position in the absence of any external load [1]. The strain measures displacements through kinematic laws. An external force applied to an elastic model induces stress while satisfying equilibrium conditions. The stress is related to the strain related through constitutive laws based on the

material's mechanical properties. The stiffness governs the relationship between the external forces and the resulting displacements at equilibrium, and is used to compute the displacements and drive the simulation of an elastic soft body.

2.2.1 Linear Elasticity

The strain is a measure of the amount of deformation in a body. In the simplest one-dimensional case, it is a measure of elongation, e.g., the ratio by which a spring elongates when subject to a certain force. It is a rate of the displacements in body relative to its reference position across normal and shearing directions. In three dimensions, the strain is a second-order tensor that measures deformations across three principal planes in three directions associated with an infinitesimal cubic element:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{00} & \varepsilon_{01} & \varepsilon_{02} \\ \varepsilon_{10} & \varepsilon_{11} & \varepsilon_{12} \\ \varepsilon_{20} & \varepsilon_{21} & \varepsilon_{22} \end{bmatrix}. \quad (2.19)$$

The stress is a measure of the intensity of forces in a body. It has normal and shearing components. In three dimensions, the stress is a second-order tensor that measures forces across three principal planes in three directions associated with an infinitesimal cubic element:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{00} & \sigma_{01} & \sigma_{02} \\ \sigma_{10} & \sigma_{11} & \sigma_{12} \\ \sigma_{20} & \sigma_{21} & \sigma_{22} \end{bmatrix}. \quad (2.20)$$

The stress and strain tensors are symmetric and can be represented as 6-dimensional vectors for convenience:

$$\boldsymbol{\varepsilon} = [\varepsilon_{00} \ \varepsilon_{11} \ \varepsilon_{22} \ 2\varepsilon_{01} \ 2\varepsilon_{12} \ 2\varepsilon_{20}], \quad (2.21)$$

$$\boldsymbol{\sigma} = [\sigma_{00} \ \sigma_{11} \ \sigma_{22} \ \sigma_{01} \ \sigma_{12} \ \sigma_{20}]. \quad (2.22)$$

The strain can be evaluated using the Green tensor as:

$$\varepsilon_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \sum_{k=0}^2 \frac{\partial^2 u_k}{\partial x_i \partial x_j} \right), \quad (2.23)$$

and for small deformations (linear elasticity), using the Cauchy tensor (a first order approximation of the Green tensor):

$$\varepsilon_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right). \quad (2.24)$$

Using the Cauchy tensor, the strain-displacement equation (kinematics) relating the strain to displacement is expressed as:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{00} \\ \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{01} \\ 2\varepsilon_{12} \\ 2\varepsilon_{20} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_0}{\partial x_0} \\ \frac{\partial u_1}{\partial x_1} \\ \frac{\partial u_2}{\partial x_2} \\ \frac{\partial u_0}{\partial x_1} + \frac{\partial u_1}{\partial x_0} \\ \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \\ \frac{\partial u_2}{\partial x_0} + \frac{\partial u_0}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x_0} & 0 & 0 \\ 0 & \frac{\partial}{\partial x_1} & 0 \\ 0 & 0 & \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_0} & 0 \\ 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_0} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = \mathbf{B}\mathbf{u}, \quad (2.25)$$

and since the internal forces in a hyperelastic solid are fully characterized by the stress, the (static) equilibrium equation relating the stress to external forces is expressed as:

$$\mathbf{B}^\top \boldsymbol{\sigma} - \mathbf{f} = \mathbf{0}. \quad (2.26)$$

The deformation is characterized by the constitutive equation that relates the stress to the strain through a (fourth-order) elasticity tensor:

$$\boldsymbol{\sigma} = \mathbf{E}\boldsymbol{\varepsilon}. \quad (2.27)$$

For isotropic elastic materials, where the material properties do not vary with direction, the elasticity tensor can be expressed using two independent parameters due to the symmetries involved: the Young's elasticity modulus, E , and Poisson's

transverse ratio, ν , as $\boldsymbol{\sigma} = \lambda \text{tr}(\boldsymbol{\varepsilon})\mathbf{I} + 2\mu\boldsymbol{\varepsilon}$:

$$\begin{bmatrix} \sigma_{00} \\ \sigma_{11} \\ \sigma_{22} \\ \sigma_{01} \\ \sigma_{12} \\ \sigma_{20} \end{bmatrix} = \begin{bmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \begin{bmatrix} \varepsilon_{00} \\ \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{01} \\ 2\varepsilon_{12} \\ 2\varepsilon_{20} \end{bmatrix}, \quad (2.28)$$

where λ and μ are the Lamé parameters given by:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}. \quad (2.29)$$

Young's elasticity modulus relates normal stress and strain while Poisson's transverse ratio relates lateral contraction to longitudinal strain. E and ν can be measured and used to accurately simulate the material's behavior.

Substituting the strain-displacement equation (2.25) into the constitutive equation (2.27) then into the static equilibrium equation (2.26) results in a set of partial differential equations in terms of \mathbf{u} :

$$\underbrace{\mathbf{B}^T \mathbf{E} \mathbf{B}}_{\mathbf{K}} \mathbf{u} - \mathbf{f} = \mathbf{0}. \quad (2.30)$$

The deformable model is simulated by (numerically) solving this system of equations using the finite element method (FEM) at every discrete step of the simulation.

2.2.2 Finite Element Method

FEM is a numerical approximation technique for computing solutions to boundary value problems defined on a domain Ω . The domain is subdivided into a mesh of simpler finite subdomains (finite elements) with simple geometric shapes, $\Omega^{[e]}$, that properly partition the domain with disjoint interiors. These conditions are

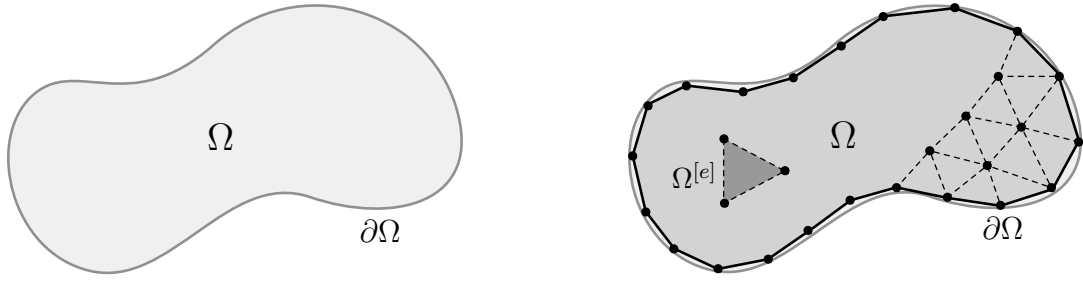


Figure 2.4 Discretization of a continuous domain Ω into a partitioning set of finite elements $\Omega^{[e]}$.

satisfied by the manifold discretization presented in §2.1.1:

$$\Omega = \bigsqcup_e \Omega^{[e]}, \quad (2.31)$$

Each element is represented by a set of nodes where each node with index i has an associated basis function, φ_i , bound to it, *i.e.* $\varphi_i(x_j) = \delta_{ij}$, and with local support inside the element, *i.e.* φ_i is non-zero only inside the element and zero elsewhere. The barycentric coordinates ξ_i (§2.1.1) defined inside an element satisfy these conditions and are used as basis functions with the linear FEM:

$$\sum_{e,i} \varphi_i^{[e]} u_i^{[e]} = \sum_e \boldsymbol{\varphi}^{[e]\top} \mathbf{u}^{[e]} = \sum_e \boldsymbol{\xi}^{[e]\top} \mathbf{u}^{[e]}. \quad (2.32)$$

The stiffness matrix and force vector can then be evaluated inside each element using (2.30) as:

$$\mathbf{K}^{[e]} = \int_{\Omega^{[e]}} \mathbf{B}^\top \mathbf{E} \mathbf{B} \, d\mathbf{x}^{[e]} = v^{[e]} \mathbf{B}^{[e]\top} \mathbf{E}^{[e]} \mathbf{B}^{[e]}, \quad (2.33)$$

$$\mathbf{f}^{[e]} = \int_{\Omega^{[e]}} \mathbf{N}^\top \mathbf{f} \, d\mathbf{x}^{[e]} = v^{[e]} \mathbf{N}^{[e]\top} \mathbf{f}, \quad (2.34)$$

where $\mathbf{E}^{[e]}$ is constant throughout the domain, $\mathbf{B}^{[e]}$ can be directly evaluated in a tetrahedral element with barycentric (linear) basis functions [40] and has the

following configuration:

$$\mathbf{B}^{[e]} = \frac{1}{6v^{[e]}} \begin{bmatrix} a_0 & 0 & 0 & a_1 & 0 & 0 & a_2 & 0 & 0 & a_3 & 0 & 0 \\ 0 & b_0 & 0 & 0 & b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 & 0 \\ 0 & 0 & c_0 & 0 & 0 & c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 \\ b_0 & a_0 & 0 & b_1 & a_1 & 0 & b_2 & a_2 & 0 & b_3 & a_3 & 0 \\ 0 & c_0 & b_0 & 0 & c_1 & b_1 & 0 & c_2 & b_2 & 0 & c_3 & b_3 \\ c_0 & 0 & a_0 & c_1 & 0 & a_1 & c_2 & 0 & a_2 & c_3 & 0 & a_3 \end{bmatrix}, \quad (2.35)$$

and the volume, $v^{[e]}$, of an element with nodes \mathbf{x}_0 , \mathbf{x}_1 , \mathbf{x}_2 , and \mathbf{x}_3 , is given by:

$$v^{[e]} = \frac{1}{6} \begin{vmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \\ 1 & 1 & 1 & 1 \end{vmatrix} \quad (2.36)$$

$$= \frac{1}{6} (\mathbf{x}_0 - \mathbf{x}_3) \cdot (\mathbf{x}_1 - \mathbf{x}_3) \times (\mathbf{x}_2 - \mathbf{x}_3). \quad (2.37)$$

The order of nodes affects the sign of the triple product and a coherent orientation of the vertices is used across the mesh ensures that the volume is positive.

For a domain discretized with n_0 nodes, \mathbf{u} and \mathbf{f} have $3n_0$ entries and \mathbf{K} has $3(n_0 \times n_0)$ entries, and are assembled using the direct stiffness method where the individual $\mathbf{K}^{[e]}$ and $\mathbf{f}^{[e]}$ from each element $\Omega^{[e]}$ are accumulated into \mathbf{K} and \mathbf{f} using their global node indices:

$$\mathbf{K} = \bigoplus_e \mathbf{K}^{[e]}, \quad (2.38)$$

$$\mathbf{f} = \bigoplus_e \mathbf{f}^{[e]}. \quad (2.39)$$

Every entry associated with an element at a node with global index i is mapped to the global entries with indices $3i$, $3i + 1$, and $3i + 2$, for rows in \mathbf{f} and rows/columns in \mathbf{K} . The assembly process along with the use of a global displacement and force vectors ensure that the conditions of displacement compatibility and force equilibrium are satisfied, *i.e.* the displacement at a node is the same in all elements and the sum of forces from all elements at a node balances out the external force at that node [31].

2.2.3 Corotational Method

The linear elastic model yields fast and stable simulations but might not be suitable for handling large deformations. This is mainly due to the inability of the first order strain measures to capture geometric non-linearities. Using the Green strain can alleviate this problem as it is invariant under rotation. However, this comes at an increased computational cost as the system has to be linearized in multiple steps where the stiffness matrix and force vector are no longer constant and must be reevaluated. This would severely impact the speed of the simulation and risk introducing numerical instabilities away from the equilibrium state [8].

The corotational method is a less computationally demanding approach that can be used with the Cauchy strain and linear FEM to capture the rotational part of the deformation gradient and mitigate the effect of geometric non-linearities. The elements are rotated back to their reference orientation along with the forces then the displacements are computed in that reference orientation and finally rotated back to the current orientation. The positions of the nodes of an element $\mathbf{x}^{[e]}$ can be computed from the positions at rest $\mathbf{x}^{[e,0]}$ using

$$\begin{bmatrix} \mathbf{x}^{[e]} \\ \mathbf{1} \end{bmatrix} = \mathbf{T}^{[e]} \begin{bmatrix} \mathbf{x}^{[e,0]} \\ \mathbf{1} \end{bmatrix}, \quad (2.40)$$

$$\mathbf{Q}^{[e]} = \mathbf{T}^{[e]} \mathbf{P}^{[e]}, \quad (2.41)$$

where $\mathbf{T}^{[e]}$ is not necessarily a rigid body transformation (there are usually stretching components) and can be computed using $\mathbf{T}^{[e]} = \mathbf{Q}^{[e]}(\mathbf{P}^{[e]})^{-1}$. The rotation part of $\mathbf{T}^{[e]}$ can be extracted using a singular value decomposition (SVD)

[8] or a polar decomposition (PD) [17]. When using SVD, $\mathbf{R}^{[e]}$ is computed using:

$$\mathbf{T}^{[e]} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \quad (2.42)$$

$$\mathbf{R} = \mathbf{V}\mathbf{U}^\top, \quad (2.43)$$

$$\mathbf{R}^{[e]} = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix}. \quad (2.44)$$

The element stiffness matrix, $\mathbf{K}^{[e]}$, and element force vector, $\mathbf{f}^{[e]}$, are updated using:

$$\mathbf{K}^{[e]} = \mathbf{R}^{[e]}\mathbf{K}^{[e]}\mathbf{R}^{[e]\top}, \quad (2.45)$$

$$\mathbf{f}^{[e]} = \mathbf{R}^{[e]}\mathbf{f}^{[e]}. \quad (2.46)$$

The main stiffness matrix has to be updated at every iteration with this method. However, the computational cost can be reduced by updating $\mathbf{R}^{[e]}$ every few steps and limiting the use of the corotational method to large enough displacements. The corotational method can be extended to handle element inversions as well [37].

2.2.4 Algebraic Constraints

The different interactions of the tissue with itself and the needle/thread are modeled by constraining the displacement vector \mathbf{u} with the aid of algebraic constraints of the form:

$$\mathbf{C}\mathbf{u} = \mathbf{b}. \quad (2.47)$$

These constraints linearly couple a subset of the nodes together, and any point in the continuum can be captured using its barycentric coordinates. Satisfying these equations is a constrained optimization problem of the form:

$$\text{minimize } \frac{1}{2}\mathbf{u}^\top\mathbf{K}\mathbf{u} - \mathbf{f}^\top\mathbf{u} \quad (2.48)$$

$$\text{with } \mathbf{C}\mathbf{u} = \mathbf{b}, \quad (2.49)$$

and can be reduced to an equivalent unconstrained problem by introducing new variables, $\boldsymbol{\lambda}$ (Lagrange multipliers), to eliminate the equality constraints. The optimal (saddle point) solution, $\{\mathbf{u}^*, \boldsymbol{\lambda}^*\}$, for this equality constrained convex quadratic minimization problem satisfies the following conditions [14]:

$$\mathbf{C}\mathbf{u}^* - \mathbf{b} = \mathbf{0}, \quad (2.50)$$

$$\mathbf{K}\mathbf{u}^* - \mathbf{f} + \mathbf{C}^\top \boldsymbol{\lambda}^* = \mathbf{0}, \quad (2.51)$$

which can be written as a system of linear equations:

$$\begin{bmatrix} \mathbf{K} & \mathbf{C}^\top \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{u}^* \\ \boldsymbol{\lambda}^* \end{Bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{b} \end{bmatrix}, \quad (2.52)$$

and is the Karush–Kuhn–Tucker system for the equality constrained problem. The values of $\boldsymbol{\lambda}^*$ are complementary to the constraints: an equality constraint either is satisfied with a corresponding non-zero multiplier or is not satisfied with a corresponding zero multiplier. The magnitude of the corresponding multiplier is an indication of how strongly a constraint affects the system and the multiplier itself represents a force in this mechanical context.

This constraint framework can be used to model simple to complex interactions between the different objects that are being simulated. It is mainly used to couple geometric positions together, namely, a vertex v can be fixed at position $\mathbf{v}^{[k]}$ by adding a constraint of the form

$$\mathfrak{g}[[v]] = \mathbf{v}^{[k]}, \quad (2.53)$$

which, when expressed in terms of displacements, is equivalent to

$$\mathfrak{g}[[v]] - \mathbf{v}^{[0]} = \mathbf{v}^{[k]} - \mathbf{v}^{[0]}. \quad (2.54)$$

The number of rows in a single constraint yields a force along a line (1 row), a plane (2 rows), or the whole space (3 rows).

2.2.5 Sparse Solvers

The stiffness matrix \mathbf{K} is symmetric by construction and is positive definite when six degrees of freedom are fixed in three dimensions:

$$\forall \mathbf{x} \neq \mathbf{0} : \mathbf{x}^\top \mathbf{K} \mathbf{x} > 0. \quad (2.55)$$

A vertex v_i can be fixed to its initial position by setting, for $3i \leq k < 3(i+1)$ and $0 \leq j < 3n_0$:

$$\mathbf{K}[k, j] = 0, \quad (2.56)$$

$$\mathbf{K}[j, k] = 0, \quad (2.57)$$

$$\mathbf{K}[k, k] = 1, \quad (2.58)$$

$$\mathbf{f}[k] = 0. \quad (2.59)$$

Given that \mathbf{K} is non-singular, the system in (2.52) is solved directly using block elimination with the Schur complement of \mathbf{K} to eliminate $\boldsymbol{\lambda}$:

$$\begin{bmatrix} \mathbf{K} & \mathbf{C}^\top \\ \mathbf{0} & -\mathbf{C}\mathbf{K}^{-1}\mathbf{C}^\top \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{b} - \mathbf{C}\mathbf{K}^{-1}\mathbf{f} \end{bmatrix}. \quad (2.60)$$

Positive definite matrices are characterized by having real positive eigenvalues and admit a unique Cholesky factorization:

$$\mathbf{K} = \mathbf{L}\mathbf{L}^\top, \quad (2.61)$$

where \mathbf{L} is a lower triangular matrix with positive diagonal entries. With this factorization, $\mathbf{K}\mathbf{u} = \mathbf{f}$ can be efficiently solved using a forward substitution followed by a backward substitution:

$$\mathbf{L}\mathbf{L}^\top \mathbf{u} = \mathbf{f}, \quad (2.62)$$

$$\mathbf{L}\mathbf{y} = \mathbf{f}, \quad (2.63)$$

$$\mathbf{L}^\top \mathbf{u} = \mathbf{y}. \quad (2.64)$$

This factorization is used for solving the reduced version of the constrained linear system in (2.60). Solving this system of equations requires a matrix product involving \mathbf{K}^{-1} , but the inverse is not computed in practice, and instead, whenever the value of a matrix product of the form $\mathbf{K}^{-1}\mathbf{x}$ is required, it is computed by solving a linear system of equations:

$$\mathbf{b} = \mathbf{K}^{-1}\mathbf{x}, \quad (2.65)$$

$$\mathbf{K}\mathbf{b} = \mathbf{x}. \quad (2.66)$$

The columns of $\mathbf{K}^{-1}\mathbf{C}^\top$ can then be computed using a sequence of substitutions, column-by-column for \mathbf{C}^\top . With $\mathbf{M} = \mathbf{C}\mathbf{K}^{-1}\mathbf{C}^\top$ and $\mathbf{q} = \mathbf{C}\mathbf{K}^{-1}\mathbf{f} - \mathbf{b}$, we end up with a dense linear system of equations:

$$\mathbf{M}\boldsymbol{\lambda} = \mathbf{q}, \quad (2.67)$$

that is solved using optimized numerical routines for dense systems [4]. The values of $\boldsymbol{\lambda}$ are then substituted in $\mathbf{K}\mathbf{u} = \mathbf{f} - \mathbf{C}^\top\boldsymbol{\lambda}$ to solve for \mathbf{u} using one last forward/backward substitution.

The stiffness and constraints matrices, \mathbf{K} and \mathbf{C} , are usually quite sparse as they are reflections of the underlying mesh connectivity. Their non-zero pattern depends on the connectivity of the mesh and the polynomial order of the interpolation functions that are used. A sparse matrix with m non-zero entries is more efficiently represented using a $3 \times m$ array where each triplet contains the row and column indices of an entry and its numerical value.

The factorization of \mathbf{K} into its Cholesky form is computed and stored using specifically tailored numerical routines and data structures from [28]. These routines and structures exploit the sparsity pattern of \mathbf{K} through a symbolic phase that determines the non-zero pattern of \mathbf{K} using an elimination tree, followed by a fill-reducing ordering of the matrix entries based on its reachability graph, and finally a recursive numerical factorization of \mathbf{K} to compute its Cholesky factors. The

rows and columns of \mathbf{K} are shuffled to reduce fill-in and a permutation matrix \mathbf{P} , a square matrix with exactly one entry of 1 in each row and column and 0 elsewhere, is used:

$$\mathbf{K} = \mathbf{P}^\top \mathbf{K} \mathbf{P}. \quad (2.68)$$

Since the topology of the mesh is constant, \mathbf{K} is factorized only once in the beginning of the simulation, after fixing the necessary number of degrees of freedom, and its factorization is reused throughout the simulation with forward/backward substitutions.

2.2.6 Collision Response

At every step of the simulation, all the soft models are checked for inter- and intra-intersections, in order to prevent or allow penetrations. The checks are performed continuously where the motion of the nodes are interpolated in space and time using techniques involving fast penetration filters coupled with a bounding volume hierarchy (BVH) [45]. Only the boundary triangular surfaces of the models, encoded on Level 3 of our mesh representation, are used when performing these collision tests, and an indirection container is used to map the two different sets of triangle indices together; the ones used in the mesh representation and the ones used in the continuous collision detection routines.

Two types of intersections are handled: intersections between a vertex and a face (vertex–face (VF)) and intersections between two edges (edge–edge (EE)), along with their corresponding timestamps. These contacts are modeled with the aid of properly imposed algebraic constraints that prevent inter- and intra-penetration by holding the collision points together. This formulation allows the underlying soft model to dictate the deformations as the constraints can either be active or inactive based on the value of their Lagrange multipliers. A constraint is active when it has a non-zero multiplier and can be further checked for whether it is pulling the two contact points closer together or away from each other.

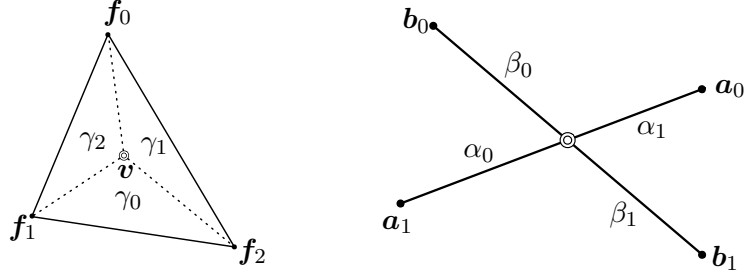


Figure 2.5 Vertex–Face and Edge–Edge intersections.

With reference to ⟨2.5⟩, a VF contact between a point with barycentric coordinates $\boldsymbol{\gamma}$ in face f and a vertex v is represented as

$$\mathbf{g}[[f]] \boldsymbol{\gamma} - \mathbf{g}[[v]] = \mathbf{0}, \quad (2.69)$$

and expands to

$$\begin{bmatrix} f_{0,x} & f_{1,x} & f_{2,x} \\ f_{0,y} & f_{1,y} & f_{2,y} \\ f_{0,z} & f_{1,z} & f_{2,z} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \end{bmatrix} - \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad (2.70)$$

and an EE contact between two points in two edges a and b with barycentric coordinates $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, respectively, is represented as

$$\mathbf{g}[[a]] \boldsymbol{\alpha} - \mathbf{g}[[b]] \boldsymbol{\beta} = \mathbf{0}, \quad (2.71)$$

and expands to

$$\begin{bmatrix} a_{0,x} & a_{1,x} & 0 \\ a_{0,y} & a_{1,y} & 0 \\ a_{0,z} & a_{1,z} & 0 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ 0 \end{bmatrix} - \begin{bmatrix} b_{0,x} & b_{1,x} & 0 \\ b_{0,y} & b_{1,y} & 0 \\ b_{0,z} & b_{1,z} & 0 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \quad (2.72)$$

These expanded forms are useful in picturing how the coefficient and node indices are extracted and stored as a list of triplets in \mathbf{C} .

⟨2.6⟩ shows an example of what can be achieved with this formulation in terms of handling of contact using a practical scenario. There are two VF and four EE collisions resulting in six constraints in total that will be encoded as 6×3 rows

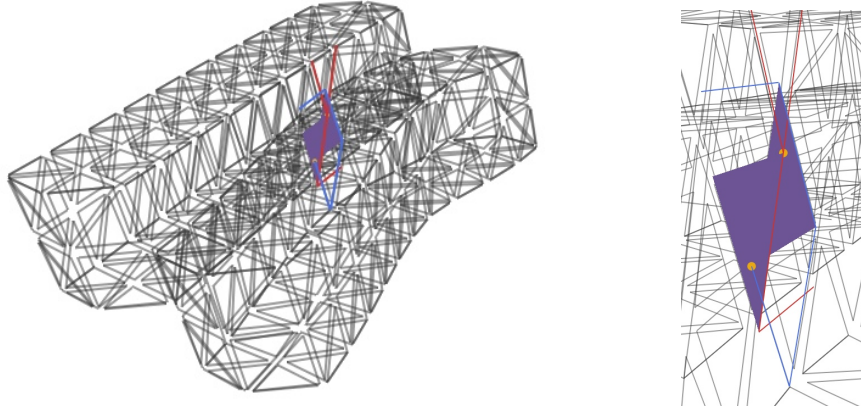


Figure 2.6 Deformation of two soft slabs, with contact handling constraints in place to prevent interpenetration. The contact complex is zoomed in the right figure (VF in *orange and purple*, and EE in *red and blue*).

in **C**. The constraints are checked as they are added to the system as not to over-constrain it; $\mathbf{CK}^{-1}\mathbf{C}^\top$ could become singular or ill-conditioned if incompatible or redundant constraints (rows) are added and the over-constrained tetrahedrons would collapse into a face, edge, or vertex, depending on the number of degrees of freedom affected when all the associated constraints have to be satisfied.

The active constraints are kept or removed dynamically based on the value of their respective Lagrange multipliers, e.g. for a VF constraint, the associated Lagrange multiplier represents the mutual force between the vertex and the face, and the constraint is removed whenever the normal component of its associated force along the supporting face becomes negative: $\hat{\mathbf{n}}^\top \boldsymbol{\lambda} < 0$; as that implies that the face and vertex are pulling away from each other.

More interesting effects can be achieved using this simple yet powerful model. As these constraints dictate that two given points be equal in position, sticking effects will be noticeable at the boundaries. In order to prevent this, the tangential components of the constraints are released to allow free tangential movement of the two boundaries with respect to each other using:

$$\hat{\mathbf{n}}^\top g[[f]] \boldsymbol{\gamma} = \hat{\mathbf{n}}^\top g[[v]], \quad (2.73)$$

where $\hat{\mathbf{n}}$ is the face normal vector. This allows the two coupled points to slide along

the plane supporting the face f . A similar equation is used for EE constraints by averaging the normal vectors of the two (boundary) faces incident to each edge. The coefficients of these equations have to be updated every time the normal vectors are updated, *i.e.* every time the geometry of the mesh is modified.

Note that different representations can be used for the different behaviors (visual, physical, and positional) of soft tissue and these different representations are usually coupled at the surface nodes or embedded within the interior cells. For example, a highly detailed surface can be used for rendering and a coarser grid for collision detection. We opt to use a single adaptive manifold mesh in this work, for all representations and behaviors, that is fine on the surface and coarser on the interior, and generated using the routines from [58, 62] and [61] for 3-manifold and 2-manifold meshes, respectively, with a spatially varying sizing field.

2.3 Related Work

The development of surgical training simulators and procedure-rehearsal systems requires reliable models for suture simulation that can pass the tests of face, construct, and content validity. The importance of this problem has led to a number of efforts for modeling various aspects of the suturing task.

An early work by Berkley et al. [13] used constraints to model suture closing in a FEM model. It presents a real-time methodology based on linear FEM analysis that is characterized by high model resolution, low processing time, unrestricted multipoint surface contact, and adjustable boundary conditions. DiMaio and Salcudean [7, 20, 21] also uses FEM models for needle insertion and steering inside soft tissue where a novel interactive virtual needle insertion simulation is presented. The insertion model simulates three-degree-of-freedom needle motion, physically-based needle forces, linear elasto-static tissue deformation and needle flexibility for the planning and training of percutaneous therapies and

procedures. A real-time simulation algorithm allows users to manipulate the virtual needle as it penetrates a tissue model, while experiencing steering torques and lateral needle forces through a planar haptic interface. An experimental system is used to validate the approach. They also present a method in [10] for estimating the force distribution that along a needle shaft during insertion is described. A two-dimensional linear elasto-static material model, discretized using FEM, is used to derive contact force information that is not directly measurable. A condensation technique is used for the simulation.

Duriez et al. [39] describe a flexible needle insertion model that does not re-mesh locally but instead uses complementarity constraints to tie points on the needle to the elements of the tissue mesh. This strategy has obvious computational savings and allows realistic biopsy and brachytherapy simulations to be performed. Laycock and Day [33] survey some early models that have been proposed for generating contact forces. More recent works have successfully used models involving linear complementarity relations between gaps and reaction forces [43] to handle contact.

A method for collision detection is used by Lee and Lee [52] where collisions between a discretized needle model and a voxel-based human tissue model with multiple layers and material properties are detected. In these works, node repositioning, additions, and local re-meshing are performed in a two-dimensional or three-dimensional volumetric FEM mesh to support the compatible sliding and sticking movements of a needle inside the mesh. Another set of research efforts by O’Leary et al. in [12] has focused on experiments intended to produce realistic in-vivo and in-vitro values for friction forces, puncture thresholds, and other parameters needed in validated simulations. Contact involving rigid tools and deformable models have been studied in physically-based surgical simulations and related haptic environments. Methods for collision detection for deformable models have been described in [26, 29, 45].

Guébert et al. [41] propose a promising simulation based on complementarity constraint modeling of all interactions between surgical threads and the embedding soft tissue. An implicit integration powered method, based on complementarity constraints, is introduced for simulating virtual sutures in soft tissue. It focuses on modeling the physical nature of the interactions between a soft anatomical structure and a needle or surgical thread during a suturing task. Puncturing through soft tissue is modeled along with friction. While in [34] asymmetric bevel-tip steerable needles are simulated using an FEM nonlinear elastic material model where the parameters are based on lab measurements. The study focuses on the sensitivity of the up forces to tissue rupture toughness and re-meshing is used. Nageotte et al. [25] present a planning method for simulating the path of circular needles using kinematic and geometric analysis to help surgeons perform a stitching task in laparoscopic surgeries.

Choi et al. [49] describes a suturing system using the physics available in a commodity physics engine including line springs and chained linear segments. Mass-spring chains have also been proposed for thread modeling. A model that uses a spline endowed with a continuum dynamic model and sliding constraints was proposed in [16]. It models the thread as a spline animated by continuous mechanics. Sliding point constraints guide the position of the thread. The direction of the thread can be constrained too. An adapted model of friction is proposed. Linear mass-spring models are used to simulate pulling sutures in [11] through a deformable model. Two separate deformable surfaces can be connected using a suture. A multi-modal environment to teach basic suturing and knotting techniques is presented in [44]. Tissue is modeled as a modified mass-spring system and the suturing material as a mechanics-based deformable linear object.

Chentanez et al. [38] presents algorithms for simulating and visualizing the insertion and steering of needles through deformable tissues for surgical training and planning. A novel algorithm for local re-meshing that quickly enforces the

conformity of the tetrahedral meshes to the curvilinear needle path coupled with an efficient method for coupling the three-dimensional finite element simulation with a one-dimensional inextensible rod with stick-slip friction. Optimizations that reduce the computation time for physically-based simulations are used too.

In [19], Crouch et al. demonstrate through collected experimental data that time- and velocity-dependent nature of the deformation resulting from needle insertion into soft tissue. The deformation during insertion is well represented using a velocity-dependent force function with a linear elastic finite element model. Goksel et al. in [22] present a three-dimensional needle-tissue interaction model that is adapted to accommodate arbitrary meshes so that the anatomy can be effectively meshed. The model is used to design a prostate brachytherapy simulator and needle-tissue coupling is achieved with node repositioning and addition. Geometric techniques and deformable models for simulating knots are presented in [5, 15, 18, 27, 30, 32].

As can be seen, suture simulation is an active field of research and has been tackled by various academic and industrial groups. The different models strive to reach a compromise that provides a balance between high-fidelity and real-time simulation. However, none of the techniques that we have reviewed explicitly decompose the motion of the needle into disjoint components or expand on the specifics of incrementally tracking a rigid needle through the underlying soft tissue tetrahedral mesh using simplicial adjacency operations. Our contribution aims to fill this gap and provide robust and efficient techniques for achieving that task.

Chapter 3

Methods

3.1 Motion Decomposition

The motion of a rigid needle can be modeled as rigid body transformations applied to the frame supporting its orientation and position. These transformations are processed at every inter-frame and can be decomposed into two components: A sliding component for motion that is tangential to the needle's axis, and a sticking component for motion that is transversal to the needle's axis.

Let the positions and orientations of the needle at two successive time steps be denoted by $\mathbf{R}^{[k]}$ and $\mathbf{R}^{[k+1]}$, respectively and $\mathbf{T}^{[k]}$ be the rigid body transformation between them, consisting of a rotation \mathbf{Q} and a translation \mathbf{t} :

$$\mathbf{R}^{[k+1]} = \mathbf{T}^{[k]} \mathbf{R}^{[k]}, \quad (3.1)$$

$$\mathbf{T}^{[k]} = \left[\begin{array}{c|c} \mathbf{Q} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array} \right]. \quad (3.2)$$

$\mathbf{T}^{[k]}$ is the result of an end-user interacting with the system at step k , and it is convenient to decompose $\mathbf{T}^{[k]}$ into two components: an axial sliding component where the needle moves along its natural axis with every point moving tangentially along the axis, and a transverse sticking component where the needle translates and rotates to reach its final position. These two components are used in updating the displacements and forces in the system in two fractional steps.

$$\mathbf{R}^{[k+1]} = \mathbf{T}^{[k]} \mathbf{R}^{[k]} = \underbrace{\mathbf{D}^{[k]}}_{\text{sticking}} \underbrace{\mathbf{S}^{[k]}}_{\text{sliding}} \mathbf{R}^{[k]}. \quad (3.3)$$

$\mathbf{S}^{[k]}$ is a transform that acts along the axis of the needle by moving it by a

certain signed amount, whereas $\mathbf{D}^{[k]}$ is a transform that moves the needle along the directions transversal to the needle's axis. As the needle is assumed to be rigid, $\mathbf{S}^{[k]}$ must be uniform, *i.e.*, all the needle points slide by the same amount and along the same direction; the needle slides as a whole. On the other hand, $\mathbf{D}^{[k]}$ is not necessarily constant for all needle points. It actually maps a point from parametric space to 3-space, which corresponds to the amount by which a needle point sticks to soft tissue at every step.

The decomposition of the needle motion and corresponding displacements into these two fractional steps is primarily motivated by the different types of needle-tissue mechanical behavior that take place when the needle displaces the embedding tissue. It is worth noting that this decomposition is different from the rotation–translation rigid body decomposition. When the needle is sliding in the tissue, it is essentially gliding unimpeded except by the (typically small) dynamic friction forces along its path. The tissue deformation induced is small since it only caused by the frictional forces tangential to the needle's path. On the other hand, when the needle is sticking to the tissue, its motion causes significant bulk deformation of the embedding tissue. Larger forces are induced because of the strain energy build up in the tissue and these forces are needed to maintain the deformed shape.

The needle would rarely slide or stick in a perfect fashion in practice but let us consider these two extreme cases: When the needle perfectly slides along its axis, the sticking component vanishes and the needle does not deform the model when cutting and friction effects are ignored; it would just drive through the mesh without deforming it. When the needle moves transversally to its sliding direction, the sliding component vanishes and the needle deforms the model without changing its location with respect to the underlying tissue mesh. As an example, consider a straight needle moving perpendicular to its axis: all of its inter-frame motion will translate into sticking motion and the needle just deforms the model without

changing its topological whereabouts.

However, in practice, both types of motion take place simultaneously during interaction, and it is convenient computationally to divide an update step into a sliding step with minimal (or no) friction and tissue deformation, and a step where the needle sticks to the tissue and deforms it in its motion. While the motion takes place, the haptic forces felt by the user are the resultants of the stresses on the needle due to both the sliding and non-sliding steps. If at any point the needle’s motion is halted, the forces felt by the user will be only the resultants of the needle contact stresses due to the non-sliding component. This is essentially a viscous model of the frictional phenomenon.

Since this decomposition is not unique, our strategy is to define the sliding component to be the transformation that cause the needle to move the closest to its final position, followed by the transformation that causes bulk deformations. Note that both factors in the decomposition include rotational and translational components. The new position of the needle’s handle (where it is being grasped at the current step) is projected onto the plane containing the previous position of the needle, and the nearest point is used as anchor point for computing $\mathbf{S}^{[k]}$. The sliding component is the portion of the needle’s axis that connects the projection of the current position of the handle to its previous position and is used for all needle points. This strategy is justified by an energetic consideration where the distance between successive positions of the handle is minimized; more refined versions of this decomposition strategy are possible.

The sliding component is represented as a scalar—a distance, σ , along the natural (whether linear or circular) sliding direction. It is computed using the inter-frame motion, the natural sliding direction of the tool, and the current support point (handle). As described previously, the current position of the handle is projected along the previous sliding direction and the amount by which the handle slides to reach its projected position defines the uniform sliding amount, *i.e.*, the

needle is treated as a whole when sliding.

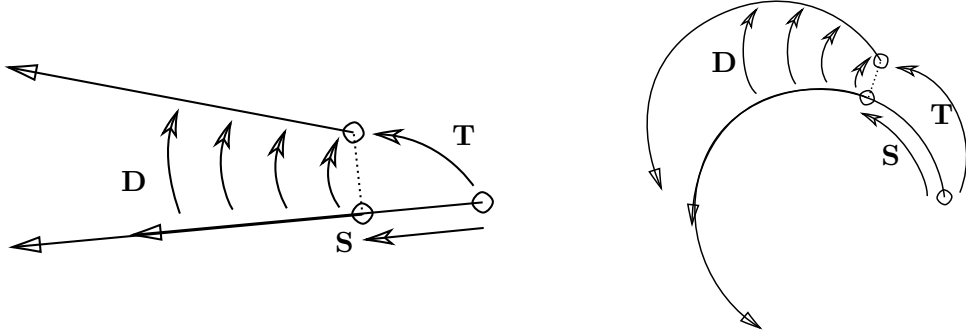


Figure 3.1 Linear and semi-circular motion decomposition.

3.1.1 Needle Parametrization

Two common types of needle geometry are handled: straight and semi-circular needles, and they are encoded using a unified parametric arc-length representation. A needle has an end, where it is usually grasped with a thread attached to it, and a tip, through which it is driven through soft tissue. The end is at parametric coordinate 0 and the tip is at ℓ (the needle's axial length). A needle then can be represented with $0 \leq \alpha \leq \ell$ as

$$\mathbf{s}(\alpha) = \mathbf{p} + \mathbf{w}(\alpha). \quad (3.4)$$

Let $\boldsymbol{\tau}^{[k]}(\alpha)$ denote the motion of a needle point between two consecutive frames k and $k + 1$ (inter-frame motion) with parametric coordinate α :

$$\boldsymbol{\tau}^{[k]}(\alpha) = \mathbf{s}^{[k+1]}(\alpha) - \mathbf{s}^{[k]}(\alpha). \quad (3.5)$$

The sliding component $\sigma^{[k]}(\alpha)$ is constant for all points $\mathbf{s}(\alpha)$ spanning the needle's length and it is represented as a signed distance (scalar) along the direction of the needle's axis:

$$\sigma^{[k]}(\alpha) \equiv \sigma^{[k]}, \quad (3.6)$$

which translates to a simple parametric coordinate shift using an arc-length needle parametrization where the shifted sliding position is denoted with $\bar{\mathbf{s}}(\alpha)$. For a point

with parametric coordinate α , $\boldsymbol{\delta}^{[k]}(\alpha)$ represents the sticking amount of the parametric point α at step k .

$$\begin{aligned}
\mathbf{s}^{[k+1]}(\alpha) &= \boldsymbol{\tau}^{[k]}(\alpha) + \mathbf{s}^{[k]}(\alpha) \\
&= \underbrace{\boldsymbol{\delta}^{[k]}(\alpha)}_{\text{sticking}} + \underbrace{\boldsymbol{\sigma}^{[k]}(\alpha)}_{\text{sliding}} + \mathbf{s}^{[k]}(\alpha) \\
&= \boldsymbol{\delta}^{[k]}(\alpha) + \mathbf{s}^{[k]}(\alpha + \sigma^{[k]}) \\
&= \boldsymbol{\delta}^{[k]}(\alpha) + \bar{\mathbf{s}}^{[k]}(\alpha)
\end{aligned} \tag{3.7}$$

For a straight needle, \mathbf{p} is the end point and $\mathbf{w}(\alpha) = \alpha \hat{\mathbf{w}}$ where $\hat{\mathbf{w}}$ is a unitary vector supporting the needle's axis:

$$\mathbf{s}(\alpha) = \mathbf{p} + \alpha \hat{\mathbf{w}}. \tag{3.8}$$

The sliding amount, $\sigma^{[k]}$, is extracted by projecting the current position of the handle, $\mathbf{s}^{[k+1]}(\bar{\alpha})$, along the previous supporting direction:

$$\sigma^{[k]} = \boldsymbol{\tau}^{[k]}(\bar{\alpha}) \hat{\mathbf{w}}^{[k]}, \tag{3.9}$$

and the transverse component, $\boldsymbol{\delta}^{[k]}(\alpha)$, is computed using:

$$\boldsymbol{\delta}^{[k]}(\alpha) = \boldsymbol{\tau}^{[k]}(\alpha) - \sigma^{[k]} \hat{\mathbf{w}}^{[k]}, \tag{3.10}$$

for every needle point as shown in [\(3.1\)](#).

The linear motion decomposition approach can be generalized for arbitrary continuous needles shapes by approximating their outline with a sequence of line segments. The motion of these line segments can then be decomposed by applying the linear motion decomposition technique to each line segment and aggregating the sliding components together into a single uniform component, which is in turn applied to the whole needle. When this technique is applied to a circular arc, we get a formulation for circular motion decomposition.

For a semi-circular needle, \mathbf{p} is the center of the needle's supporting circle with radius r :

$$\mathbf{s}(\alpha) = \mathbf{p} + r \cos\left(\frac{1}{r}\alpha\right) \hat{\mathbf{u}} + r \sin\left(\frac{1}{r}\alpha\right) \hat{\mathbf{v}}, \tag{3.11}$$

with $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ two unitary vectors orthogonal to each other and $\hat{\mathbf{w}}$ a unitary vector supporting the direction that joins the needle's center to its end.

Circular decomposition follows directly from the limit case when approximating a circular needle by a sequence of line segments. Two orthonormal frames are built and the amount of rotation between them is approximated along the closest point to the handle. The handle, $\mathbf{s}^{[k+1]}(\bar{\alpha})$ is projected along the previous (circular) direction using:

$$\mathbf{c} = \mathbf{s}^{[k+1]}(\bar{\alpha}) - \mathbf{p}^{[k]}, \quad (3.12)$$

$$\hat{\mathbf{q}} = \text{normalize}(\mathbf{c} - (\mathbf{c}^\top \hat{\mathbf{n}})\hat{\mathbf{n}}), \quad (3.13)$$

$$\mathbf{z} = \mathbf{p}^{[k]} + r\hat{\mathbf{q}}, \quad (3.14)$$

where $\vec{\mathbf{n}} = \hat{\mathbf{u}} \times \hat{\mathbf{v}}$. \mathbf{z} is the projection of $\mathbf{s}^{[k+1]}(\bar{\alpha})$ along the previous direction and is computed as the closet point to the needle handle's previous position. With

$$\hat{\mathbf{a}} = \text{normalize}(\mathbf{s}^{[k]}(\bar{\alpha}) - \mathbf{p}^{[k]}), \quad (3.15)$$

$$\hat{\mathbf{b}} = \text{normalize}(\mathbf{z} - \mathbf{p}^{[k]}), \quad (3.16)$$

the sliding amount is extracted using:

$$\sigma^{[k]} = \text{sgn}[\hat{\mathbf{n}}^\top(\hat{\mathbf{a}} \times \hat{\mathbf{b}})]r \arccos(\hat{\mathbf{a}}^\top \hat{\mathbf{b}}), \quad (3.17)$$

an amount by which the needle's handle rotated around its center in the same supporting plane between the two steps. An example using this decomposition strategy is shown in [⟨3.1⟩](#).

This parametric representation can be used to seamlessly switch between the two geometries and decouple the needle's resolution from the soft tissue mesh's resolution. It also allows the coupling of needle points to mesh points through matching parametric and barycentric coordinates. With this representation, the needle can be easily moved around based on the end-user interactions with the system. The rigid body motions resulting from the end-user interactions are directly

applied to the points and vectors supporting the needle’s representation at the handle and represented as an affine transformation matrix $\mathbf{T}^{[k]}$. The two frames, end-user devices and needle, are interlocked, and the motion is relayed between them:

$$\mathbf{s}^{[k+1]}(\alpha) = \mathbf{T}^k \mathbf{s}^{[k]}(\alpha) = \mathbf{T}^{[k]} \mathbf{p}^{[k]} + \mathbf{T}^{[k]} \mathbf{w}^{[k]}(\alpha). \quad (3.18)$$

This representation is also used to generate the visual model of the needle. The parametric space is sampled at regular intervals and a sequence of cylinders is built around the supporting curve (skeleton) to generate a visual representation of the needle. This discrete polygonal representation can be used as a collision model if more realistic interactions between the needle and soft tissue are desired.

3.2 Needle Driving

A needle’s main purpose is to introduce a thread into soft tissue to tighten a suture. It first punctures the boundary (from the outside to the inside) then drives through the tissue, paving the way for the thread to follow, and finally punctures the boundary (from the inside to the outside) and exits through. As a first step in modeling this task is to decompose the motion of the needle into two disjoint components and handle each separately.

Decomposing needle displacement into two disjoint components (sliding and sticking) allows a needle-tissue interaction simulation step to be divided into two disjoint phases: a bookkeeping (sliding) phase for tracking the needle inside the mesh, and a deformation (sticking) phase. Tracking the needle as it moves inside the soft tissue is an integral component of the suture simulation. It consists of keeping track of the intersection points between the needle and the discretized tissue model, and coupling these points through the use of linear constraints. The end goal is to efficiently and robustly simulate the driving of a needle into soft tissue. The different steps of our approach are depicted in <3.2>.

The most computationally expensive part of the simulation is solving the underlying system of linear equations, nonetheless, other tasks have a considerable computational cost as well. Therefore, computational savings across any level are as equally important. Our strategy relies on using a list of history cells and incrementally updating it using the sliding component and the simplicial adjacency operators provided by the supporting data structure in order to generate the constraints representing transverse motion.

The uniform non-deforming component is used in updating the history list of visited cells which is represented using a list. This list of cells is used for incrementally computing the intersection of the needle with the deformable model and for generating the constraints that model the interaction between the needle and soft tissue. The search for the new location of the tip is initiated with the cell that contains it at the current step and a local search, guided by the sliding component, through the adjacent cells is performed to track the movement of the tip and update its position inside the mesh.

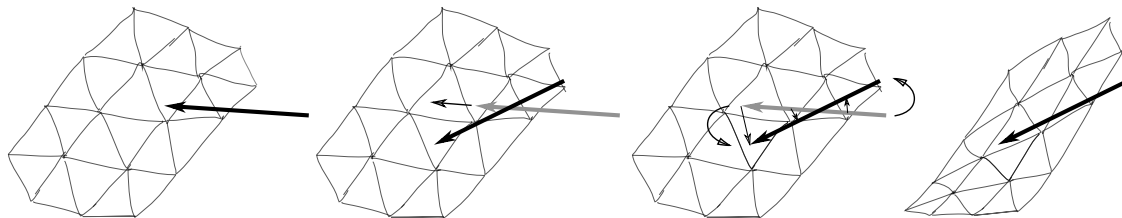


Figure 3.2 An example of tracking a straight needle inside a soft tissue mesh that depicts the general idea behind our proposed approach. Inter-frame needle motion is decomposed into two separate components: sliding (non-deforming) and sticking (deforming), then the needle is slid and its position in the mesh is updated. The new needle position is coupled to the previous cell intersections and the system is solved and updated.

3.2.1 Incremental Tracking

A straightforward way of tracking the needle inside tissue is to compute, at every inter-frame, its intersection with every cell in the discretized model. This is

clearly not suitable for real-time simulation as it comes with a heavy computational cost: one has to go over all the cells and check whether the path of the needle intersects with one of their faces. A simple and efficient method that makes use of the rigid and continuous nature of the needle to reduce the overall computational cost is needed.

Our method relies on incrementally updating the position of the needle with respect to the mesh it intersects. As the needle is represented as a continuous model, the body of the needle will follow the same topological path travelled by its tip (follow-the-leader approach) and the path of the needle tip can be used to track the entire needle and update the current intersection points. As described in §3.1, the sliding and sticking displacements are updated at every inter-frame; however, for the purpose of tracking, only the sliding component affects the intersection points (topological elements). Therefore, only the sliding information is used to update the position of the needle with respect to the models it intersects.

3.2.1.1 Tip Tracking

As the tip is moving across the simulation domain, its location with respect to the tissue’s mesh is updated by tracking the index of the cell t it resides in. This index is guaranteed to be unique because the interiors of the cells do not intersect in our representation. The exterior of the mesh is represented as a single cell, referred to as the outside cell, indexed using \perp (the outside of the mesh), and is incident to all boundary faces. This representation simplifies the process of locating the tip: it is always located in a cell, whether inside or outside the mesh, and tracking it simplifies to detecting whether it moves through a face or not, resulting in two outcomes at every step: the tip crosses a face (jumps to a different cell) or it does not (remains in the same cell). This process is repeated until the tip reach its current position.

When the tip is inside the mesh, the cell into which it moves, if any, can be

Algorithm 3.1 Tip tracking through a tetrahedral mesh.

```

1: procedure LookAround( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, t$ )
2:   if  $t \neq \perp$  then
3:     for  $f$  in  $r_{32}[[t]]$  do
4:       if  $(\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}) \cap g[[f]]$  then
5:         StepPoint( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, t, f$ )
6:     else
7:        $f \leftarrow (\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}) \cap \partial\Omega$ 
8:       if  $f \neq \perp$  then
9:         StepPoint( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, t, f$ )

10: procedure StepPoint( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, t, f$ )
11:   if  $t \neq \perp$  then
12:      $t \leftarrow [\frac{1}{4} \text{opp}[[f]]]$ 
13:   else
14:      $t \leftarrow [\frac{1}{4} f]$ 
15:    $\mathbf{s}^{[k]} \leftarrow (\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}) \cap g[[f]]$ 
16:   LookAround( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, t$ )

```

identified by jumping into the opposite of the intersected half-face and then querying the data structure for the parent cell incident to that face. Now that the cell into which the tip has moved is identified, the portion of the displacement contained within the previous cell is disregarded; this will incrementally consume the displacement vector as the tip moves along and eventually the lookup process terminates.

The lookup process is summarized in {3.1} with the tip at position $\mathbf{s}^{[k]}$ located in cell t and moving to position $\mathbf{s}^{[k+1]}$. The process can be further simplified by overloading $r_{32}[[\perp]]$ to return all boundary faces, resulting in {3.2}. These routines are capable of handling multiple boundary crossings and forward/backward motion along a sliding direction during the same inter-frame. A sample application of this routine is shown in <3.3> where a point is being tracking through a tetrahedral slab.

If the needle is purely sliding, then $\mathbf{s}^{[k+1]}(\alpha) = \bar{\mathbf{s}}^{[k]}(\alpha)$, and that fact can be used to abstract away the underlying geometry of the needle from the tracking

Algorithm 3.2 Simplified tip tracking through a tetrahedral mesh.

```

1: procedure LookAround( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, t$ )
2:   for  $f$  in  $r_{32}[[t]]$  do
3:     if  $(\overline{\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}}) \cap g[[f]]$  then
4:       StepPoint( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, t, f$ )

5: procedure StepPoint( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, t, f$ )
6:    $\overline{\mathbf{s}^{[k]}} \leftarrow (\overline{\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}}) \cap g[[f]]$ 
7:   if  $t \neq \perp$  then
8:     LookAround( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, [\frac{1}{4}\text{opp}[[f]]]$ )
9:   else
10:    LookAround( $\mathbf{s}^{[k]}, \mathbf{s}^{[k+1]}, [\frac{1}{4}f]$ )

```

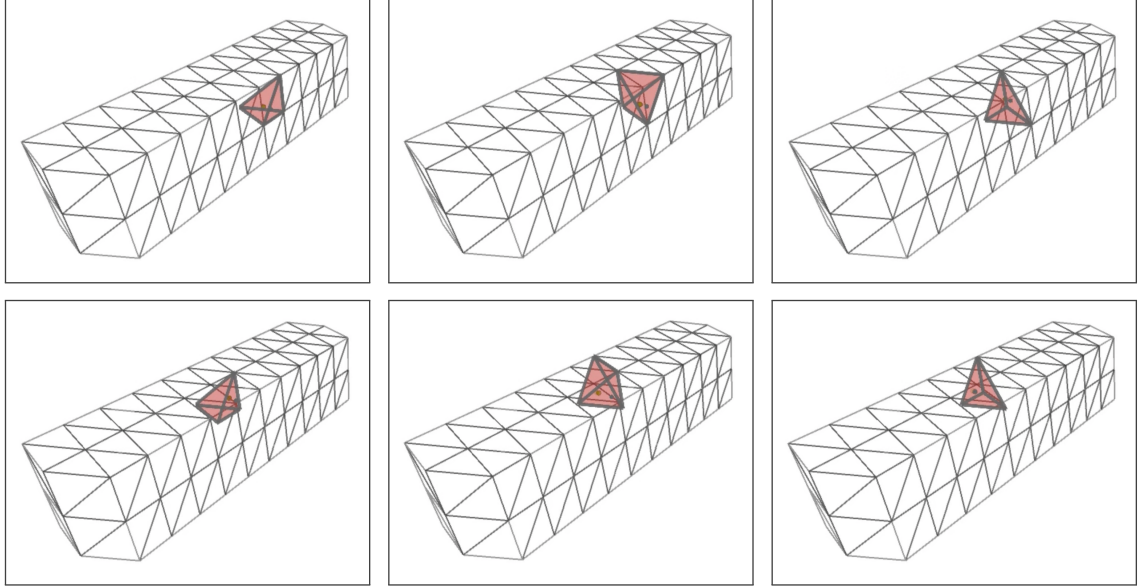


Figure 3.3 Tip tracking through a tetrahedral slab.

process. Both straight and semi-circular needles are treated as a single generic parametrized model, and the intersection routine is modified to correctly capture the intersections for each case.

Every valid intersection results in a pair of parametric/barycentric coordinates for the intersection point, α and $\boldsymbol{\xi} = (\xi_0, \xi_1, \xi_2)$, such that

$$\overline{\mathbf{s}}(\alpha) = g[[f]] \boldsymbol{\xi}, \quad (3.19)$$

and this pair of coordinates, $[\alpha, \boldsymbol{\xi}]$, is used to couple the needle and soft tissue through specific complementarity constraints. It should be noted that this approach

is not tightly coupled to the underlying data structure and supporting element and can be adapted to other mesh elements as long as the required adjacency operators are supported.

3.2.1.2 Intersection Tests

The intersection test for the tip path with a given face boils down to whether valid values can be assigned to barycentric face coordinates and needle parametric coordinates such that they coincide at a single point. It is performed in two stages: The first stage consists of sliding along the needle axis while checking whether it intersects with the plane in which the triangle (face) in question resides and the second step consists of trying to assign valid barycentric coordinates to the intersection point, if any, such that it lies inside triangle.

For a straight needle, the Möller-Trumbore's algorithm [3] is an efficient way to detect intersections between a line segment and a triangle. The intersection point satisfies $\mathbf{s}(\alpha) = g[[f]] \boldsymbol{\xi}$ which expands to $\mathbf{p} + \alpha \hat{\mathbf{w}} = [\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_3] \boldsymbol{\xi}$ and by rearranging the terms we obtain the following system:

$$[-\hat{\mathbf{w}}, \mathbf{f}_1 - \mathbf{f}_0, \mathbf{f}_2 - \mathbf{f}_0] \begin{bmatrix} \alpha \\ \xi_1 \\ \xi_2 \end{bmatrix} = \mathbf{p} - \mathbf{f}_0, \quad (3.20)$$

which is then solved using a sequence of triple products for α , ξ_1 , and ξ_2 , with $\xi_0 = 1 - (\xi_1 + \xi_2)$. An intersection is valid when $\ell \leq \alpha \leq \ell + \sigma$ and $0 \leq \xi_i \leq 1$.

The intersection test is a bit more involved for the semi-circular case. Let $\hat{\mathbf{n}}$ denote the unit normal of the face f . The plane supporting f can be implicitly represented as $(\mathbf{s} - \mathbf{f}_0)^\top \hat{\mathbf{n}} = 0$ and $\mathbf{s}(\alpha)$ has to satisfy the plane's equation if the semi-circular arc were to intersect with the triangle, *i.e.*

$$\left(\mathbf{p} + r \cos\left(\frac{1}{r}\alpha\right) \hat{\mathbf{u}} + r \sin\left(\frac{1}{r}\alpha\right) \hat{\mathbf{v}} - \mathbf{f}_0 \right)^\top \hat{\mathbf{n}} = 0. \quad (3.21)$$

By rearranging the terms of (3.21), we obtain the following identity:

$$r \cos\left(\frac{1}{r}\alpha\right) \hat{\mathbf{u}}^\top \hat{\mathbf{n}} + r \sin\left(\frac{1}{r}\alpha\right) \hat{\mathbf{v}}^\top \hat{\mathbf{n}} = -(\mathbf{p} - \mathbf{f}_0)^\top \hat{\mathbf{n}}, \quad (3.22)$$

which by a change of variables is equivalent to $a \cos \theta + b \sin \theta = c$. With $q = \sqrt{a^2 + b^2}$, the previous equation can be expressed as

$$(a/q) \cos \theta + (b/q) \sin \theta = c/q; \quad 0 \leq |(a/q)|, |(b/q)| \leq 1. \quad (3.23)$$

Since $(|a|, |b|, q)$ is a Pythagorean triplet, we can substitute (a/q) with $\sin \phi$ and (b/q) with $\cos \phi$ and the result is

$$\sin \phi \cos \theta + \cos \phi \sin \theta = \sin(\phi + \theta) = c/q. \quad (3.24)$$

If $|c/q| > 1$, the equation has no solutions and the triangle and arc do not intersect, otherwise, ϕ can be extracted using $\arctan(a/q, b/q)$ and θ is computed using

$$\theta = [\arcsin(c/q), \pi - \arcsin(c/q)] - \phi, \quad (3.25)$$

then the face barycentric coordinates $\boldsymbol{\xi}$ are computed for these two candidate points and with $\alpha = r\theta$ checked for validity when $\ell \leq \alpha \leq \ell + \sigma$, and $0 \leq \xi_i \leq 1$.

3.2.2 History Tracking

The successive tissue intersections points that are crossed by the tip can be tracked through the use of a history list container, \mathbb{H} , where \mathbb{H}_i is a combination of a cell index, t_i , a needle parametric coordinate, α_i , and a tissue barycentric coordinate, $\boldsymbol{\xi}_i$. As the tip can only switch between cells by crossing a face in a 3-manifold, the sequence of crossing events (valid intersection tests) is logged and used to populate the history list for tracking the whole body of the needle inside the tissue. A cell is added and kept in the list as long as the needle is intersecting it since operations that would change the topology of the mesh are ignored.

The history list will contain at every step only the current intersection points of the needle with the soft tissue mesh and these points are used to impose

constraints for coupling the needle and tissue together and modeling their interaction. This list of intersection points encodes a sequence of tissue cells that are adjacent through opposite faces; this is an invariant property of the history list and all operations that modify the list should maintain this property. The cells of the history list are a 2-connected subset of the underlying simplicial complex [9].

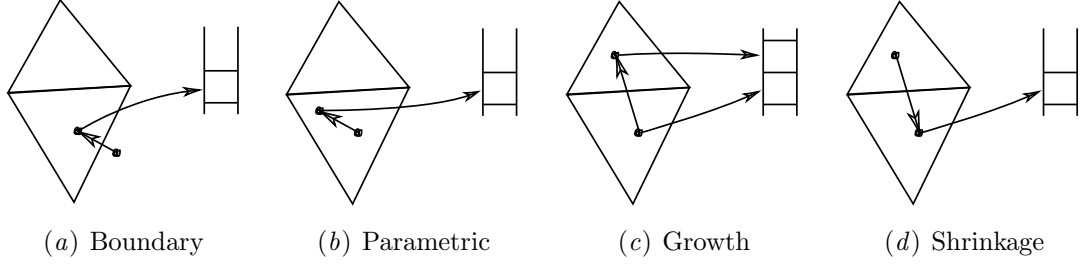


Figure 3.4 Four cases for addition, removal, and update of history cells.

The history list is treated as a stack where the top element always references the cell that contains the tip at the current step k and is used as a starting point in the search routine for the new location of the tip; \mathbb{H} is never empty and is initialized with the outside cell (\perp) at the beginning of the simulation. This search will result in four different possibilities pictured in [3.4](#) for updating the history list.

We distinguish between three sliding situations across the needle's axis: (a) forward: $\sigma^{[k]} > 0$, (b) backward: $\sigma^{[k]} < 0$, and (c) none: $\sigma^{[k]} = 0$. The sliding phase has no effect on the coupling when the needle does not slide, and the history list remains unchanged. If the needle slides forward, the path of the sliding tip is tested for intersections with the mesh and cells are added for valid intersections [3.3](#). No intersections tests are performed when the needle is sliding backward, and the cells are removed when their parametric coordinates go beyond the range of the needle's support. The barycentric coordinates are fixed throughout their lifetime in the history list as the topology of the tissue mesh does not change.

After locating the tip inside the mesh using its new position, \mathbb{H} is traversed to update the parametric coordinates of the needle points and to mark cells that are

Algorithm 3.3 History tracking with forward sliding through a tetrahedral mesh.

```

1: procedure LookAround( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}$ )
2:   for  $f$  in  $r_{32}[\mathbb{H}.top]$  do
3:      $\{\alpha, \boldsymbol{\xi}\} \leftarrow (\mathbf{s}^{[k]}(\ell), \mathbf{s}^{[k]}(\ell + \sigma^{[k]})) \cap g[f]$ 
4:     if  $\boldsymbol{\xi} \neq \mathbf{0}$  then
5:       StepPoint( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}, \{f, \alpha, \boldsymbol{\xi}\}$ )

6: procedure StepPoint( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}, \{f, \alpha, \boldsymbol{\xi}\}$ )
7:    $\sigma^{[k]} \leftarrow \sigma^{[k]} - (\alpha - \ell)$ 
8:   if  $\mathbb{H}.top \neq \perp$  then
9:      $t \leftarrow \lfloor \frac{1}{4} opp[f] \rfloor$ 
10:  else
11:     $t \leftarrow \lfloor \frac{1}{4} f \rfloor$ 
12:   $\mathbb{H}.push(\{t, \alpha, \boldsymbol{\xi}\})$ 
13:  LookAround( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}$ )

```

no longer being intersected by the needle, namely those that are outside the parametric range $[0, \ell]$. The parametric coordinates are updated at this step by subtracting the sliding component $\sigma^{[k]}$ using

$$\alpha_i^{[k+1]} = \alpha_i^{[k]} - \sigma^{[k]}. \quad (3.26)$$

If $\alpha_i^{[k+1]} > \ell$, the point lies beyond the tip and is no longer part of the needle; the corresponding entry is removed from the history list. We keep the entries with $\alpha_i^{[k+1]} < 0$ as they are used to track the thread through the tissue, however, they do not contribute to coupling the needle and tissue together. The thread is attached at $\alpha = 0$ and has a parametric range of $[0, -\eta]$ where η is the total length of the thread. The entries with $\alpha_i^{[k+1]} < -\eta$ are removed.

Boundary faces require a special treatment as they induce a deformation. Whenever the tip goes through a boundary face, the tracking process is halted as long as the tip is in contact with the boundary face to allow for surface deformations to occur; the deformation is enforced through transverse deformations and is handled in §3.2.3. The tracking resumes only after the tip punctures through the boundary or loses contact with it.

One last modification is required to properly handle inside-out punctures.

Algorithm 3.4 History tracking with forward/backward sliding through a tetrahedral mesh.

```

1: procedure TrackNeedle( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}$ )
2:   if  $\sigma^{[k]} > 0$  then
3:     LookAround( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}$ )
4:   if  $\sigma^{[k]} \neq 0$  then
5:     UpdateParametric( $\mathbb{H}, \sigma^{[k]}$ )

6: procedure LookAround( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}$ )
7:   for  $f$  in  $r_{32}[\mathbb{H}.top]$  do
8:      $\{\alpha, \boldsymbol{\xi}\} \leftarrow (\mathbf{s}^{[k]}(\ell), \mathbf{s}^{[k]}(\ell + \sigma^{[k]})) \sqcap g[f]$ 
9:     if  $\boldsymbol{\xi} \neq \mathbf{0}$  then
10:      StepPoint( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}, \{f, \alpha, \boldsymbol{\xi}\}$ )

11: procedure StepPoint( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}, \{f, \alpha, \boldsymbol{\xi}\}$ )
12:    $\sigma^{[k]} \leftarrow \sigma^{[k]} - (\alpha - \ell)$ 
13:   if  $\mathbb{H}.top \neq \perp$  then
14:      $t \leftarrow \lfloor \frac{1}{4} \text{opp}[f] \rfloor$ 
15:   else
16:      $t \leftarrow \lfloor \frac{1}{4} f \rfloor$ 
17:    $\mathbb{H}.push(\{t, \alpha, \boldsymbol{\xi}\})$ 
18:   LookAround( $\mathbb{H}, \mathbf{s}^{[k]}, \sigma^{[k]}$ )

19: procedure UpdateParametric( $\mathbb{H}, \sigma^{[k]}$ )
20:   for  $i = \mathbb{H}.length - 1$  to  $1$  do
21:      $\mathbb{H}[i].\alpha \leftarrow \mathbb{H}[i].\alpha - \sigma^{[k]}$ 
22:     if  $\mathbb{H}[i].\alpha > \ell$  then
23:        $\mathbb{H}.pop()$ 
24:     else if  $\mathbb{H}[i].\alpha < 0$  then
25:        $\mathbb{H}[i].deactivate()$ 
26:     else
27:        $\mathbb{H}[i].activate()$ 

```

Following an inside-out puncture, the tip is located outside the mesh and the last puncture point is lost in the process as the barycentric coordinates are not valid for the outside cell (\perp). This boundary face index can be recovered by keeping track of the intersected face indices instead of the cell indices; the cell indices can always be recovered using $t = \lfloor \frac{1}{4}f \rfloor$ to be used in the tracking routines.

It is worth mentioning that this whole process automatically adapts to the soft tissue mesh resolution. Finer meshes result in more intersection tests, more needle-tissue coupling points, and a more accurate needle-tissue interaction overall.

3.2.2.1 Special Cases

As the inter-frame path of the tip is being tested for collision with the boundary, one has to take into account that multiple boundary intersections can occur. When that is the case, the intersections have to be sorted (in time) based on their parametric coordinates and only the first such intersection is taken into account; the rest are discarded and will be retrieved at a later stage when the tip reaches its final position.

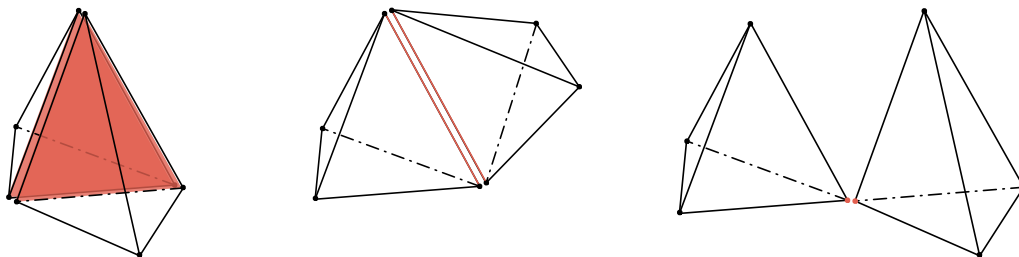


Figure 3.5 Three possible inter-cell crossings: through a face (*left*), through an edge (*middle*), and through a vertex (*right*).

As the tip exits the current cell it currently resides in, it necessarily goes through its boundary. We have assumed in our treatment throughout that the tip always exits through a face, which is not technically incorrect as the cell bound by faces, however, when that exit is through a vertex or an edge (that are part of that

face) the opposite relationship between the face that the tip exists through and the face that the tip enters through is no longer valid and the tracking process will fail; two adjacent faces can share one, two, or three vertices as shown in $\langle 3.5 \rangle$.

A remedy to this problem is to use the star of faces (r_{02} for a vertex and r_{12} for an edge; depicted in $\langle 2.3 \rangle$) as intersection candidates as the tip is guaranteed to be contained within that star; the star of faces completely wraps around the vertex or edge. In practice, the link of faces is used instead of the star to prevent a cell from being added twice to the list. Using the face link query also preserves the invariant adjacency property of the history list such that two consecutive cells always intersect, either through a vertex, an edge, or a face. An (extreme) example where the tip only intersects with edges is shown in $\langle 3.6 \rangle$.

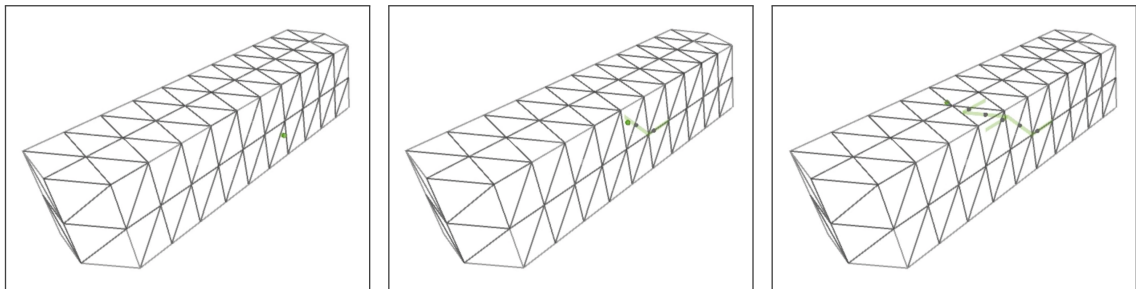


Figure 3.6 Tip tracking through a tetrahedral slab with edge intersections. The face link query is used at every single step to get list of faces to be tested for intersection.

3.2.3 Boundary Puncture

The puncturing dynamics can be simplified to a VF intersection test where the corresponding constraint is applied as described in §2.2.6. A strain-based test is used to decide whether the constraint should be released or not. The tip will deform the model until it is either pulled back and loses contact with the surface, or pushed in until it penetrates the surface. The mutual force between the tip and the boundary is extracted from the VF constraint's Lagrange multiplier, then projected onto the needle's axis.

The resulting scalar value is compared against a tissue-dependent (stress) threshold to decide whether the constraint should be released or kept active. This holds the tip and boundary point together while the tip pushes on the soft tissue. The constraint is kept active and updated until the induced force exceeds the specified threshold or the needle is pulled back and the tip is no longer engaging with the boundary. When the force exceeds the specified threshold, the constraint is released and the tip punctures the surface, driving through the mesh, crossing one or more cells before reaching its final position. When the force induced becomes negative, the constraint is released and the tip loses contact with the boundary.

3.2.3.1 Motion Playback

After releasing the constraint that holds the tip and boundary together, the boundary is going to snap back to its position on first contact with the tip, but the needle remains in the same position. The tip could have crossed multiple cells after puncturing the boundary. To detect these cells, the motion of the needle is logged from the first point of contact with the boundary to the last point of contact, up until the constraint is released.

After the playback, the needle is back to its pre-playback position and the cells crossed by the needle are part of the history list. From there on, the regular tracking procedure is applied. This motion playback process is applied whenever the tip enters or exits soft tissue, and the underlying algebraic system is solved at every step of the playback phase to handle the resulting sticking motion.

The motion of needle is logged as rigid body transformations, \mathbf{T}_i , that were applied to the frame supporting its orientation and position while its tip was interacting with the boundary. These transformations are used to replay the motion of the needle, starting with the position of initial contact with the surface, and ending with the current position; the displacements are discarded when the needle is no longer in contact with the surface. The needle is tracked inside the mesh using

the same techniques described earlier, except that there is no puncturing threshold during this phase, *i.e.*, the needle crosses boundary cells without any resistance.

After the tip punctures the boundary and the transformations are applied in reverse order (3.27), the needle moves back to its original position when it first came into contact with the boundary by applying the following transformation:

$$\mathbf{T}_0^{-1}\mathbf{T}_1^{-1}\mathbf{T}_2^{-1}\cdots\mathbf{T}_{m-1}^{-1}. \quad (3.27)$$

In practice, we store the position and orientation of the needle every time it comes in contact with the boundary and use them to restore it to its initial position at contact instead of inverting these matrices. After the needle is back to its position on first contact, the \mathbf{T}_i are decomposed into $\mathbf{D}_i\mathbf{S}_i$ successively, and the history list is updated accordingly.

⟨3.7⟩ shows an example where the tip intersects multiple cells after crossing the boundary. The mesh also deforms due to sticking, and the resultant force are rendered to the user. Now that the needle is back to its pre-playback position and the cells crossed by the needle are part of the history list, the invariant property of the history list is restored and normal tracking resumes. The motion playback routine is applied whenever the tip enters or exits soft tissue, and the underlying algebraic system is solved at every step of the playback phase to handle the resulting sticking motion. Although the playback phase requires solving the algebraic system multiple times in a row, it is only activated when interacting with the boundary.

3.2.4 Transverse Deformation

The transverse (sticking) phase affects the geometry of the mesh where the intersection points are driven into their final position (on the needle) by imposing proper algebraic constraints. $\mathbf{D}^{[k]}$ from (3.3) is basically applied to the parametric coordinates of the history list to compute the amount of sticking deformation for every coupling point. The sticking amounts are incorporated into VF constraints

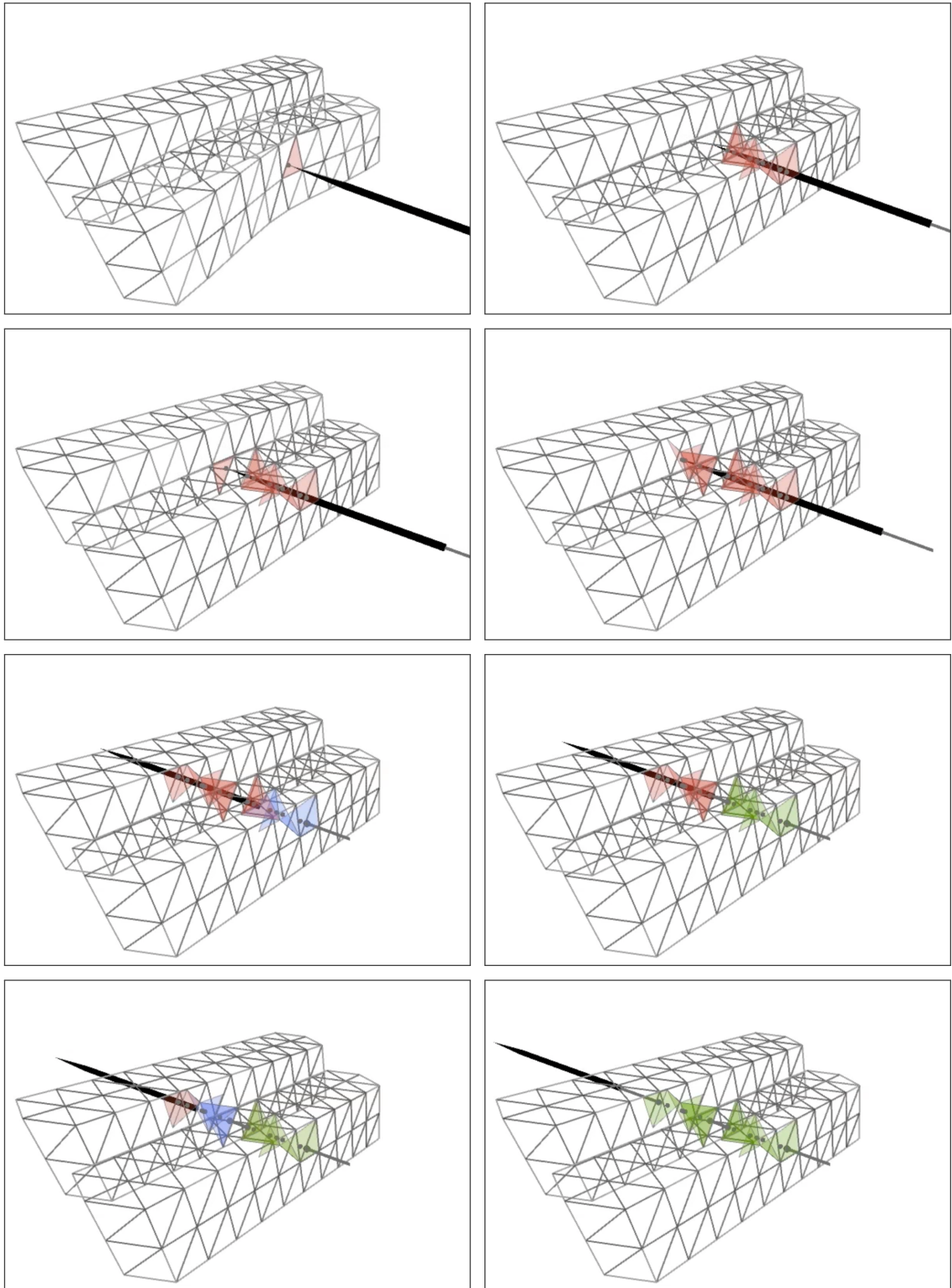


Figure 3.7 Driving a needle through two disjoint slabs. The needle punctures through four boundary faces to reach its final position. The red cells are actively tracking the needle and the blue/green cells are tracking the thread..

that guide the deformation. For every entry indexed by i in the history list, a constraint of the following form is imposed with reference to (3.19):

$$g[f_i] \boldsymbol{\xi}_i = \bar{\mathbf{s}}^{[k]}(\alpha_i) + \boldsymbol{\delta}^{[k]}(\alpha_i), \quad (3.28)$$

after which the internal forces, for each cell, can be computed using the associated Lagrange multipliers.

One trick to accelerate the simulation is to aggregate the sticking components between steps and apply the deforming constraints with the aggregate values every few steps. This strategy reduces the number of solutions required and considerably accelerates the computation time. One can even aggregate all the transformations applied to the needle between a given number of steps if increased speedup is desired, on the expense of reduced accuracy.

3.2.4.1 Slipping and Friction

The sliding phase also modifies the underlying geometry when the needle is in contact with the mesh's boundary. This happens when the tip is entering the mesh (outside-in puncture) or exiting the mesh (inside-out puncture) and is apparent in particular for inside-out punctures: the interior points coupling the needle to the mesh will stick to the needle's axis, preventing it from sliding through, and resulting in an abnormal bulge at the puncture point.

This situation requires a special treatment to allow the needle to freely slide through the (internal) contact points while it is pushing the boundary and is achieved by projecting the constraints with these points along the needle's axis, freeing one of their degrees of freedom and allow the needle to move freely along its axis through these tissue points. The resulting projected constraints are expressed for an entry at index i as:

$$\mathbf{P}_i g[f_i] \boldsymbol{\xi}_i = \mathbf{P}_i \mathbf{s}^{[k+1]}(\alpha_i), \quad (3.29)$$

where $\mathbf{P}_i = [\hat{\mathbf{u}}_i^\top; \hat{\mathbf{v}}_i^\top]$ is a 2×3 transformation matrix that projects along the plane orthogonal to the needle's axis at position α_i ; it is the same for all points for a straight needle. The constraint coupling the needle tip to the boundary remains the same. \mathbf{P} 's supporting vectors, $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$, along a needle axis $\hat{\mathbf{w}}$ are computed using:

$$\hat{\mathbf{u}} = \hat{\mathbf{a}} \times \hat{\mathbf{w}} \quad (3.30)$$

$$\hat{\mathbf{v}} = \hat{\mathbf{w}} \times \hat{\mathbf{u}} \quad (3.31)$$

with $\hat{\mathbf{a}} \in \{\pm\hat{\mathbf{w}} \times \hat{\mathbf{e}}_x, \pm\hat{\mathbf{w}} \times \hat{\mathbf{e}}_y, \pm\hat{\mathbf{w}} \times \hat{\mathbf{e}}_z\}$ with $\hat{\mathbf{e}}_x = (1, 0, 0)$, $\hat{\mathbf{e}}_y = (0, 1, 0)$, and $\hat{\mathbf{e}}_z = (0, 0, 1)$. These vectors are tested in sequence until a proper orthonormal frame is built.

After solving the system with the projected 2-row constraints, the parametric coordinates of the cells, α_i , have to be updated to match the new position of the needle with respect to the mesh since there has been some sliding. The barycentric coordinates remain the same because the needle only slides through the cells. The intersection of each history cell with the needle representation is computed with the same routines from §3.2.1.2 using the whole parametric range of the needle of the needle and the parametric coordinates are set accordingly. This is an intermediary step performed only when the needle tip is interacting with the boundary and the whole process is succinctly summarized in {3.5}.

3.3 Thread Pulling

The needle path inside the body serves to define the topological path of the thread in the mesh. The mesh elements crossed by the needle, and the puncture points introduced while entering and exiting the mesh, define an ordered data structure that is used to model the thread-tissue interaction while the thread is pulled to close an opening.

The history list with its needle parametric/tissue barycentric is used to guide the thread through the soft tissue using a follow-the-leader approach. As the

Algorithm 3.5 Different steps of needle driving.

```
1: procedure NeedleDriving( $\Omega, \mathbb{H}, \sigma$ )
2:   DecomposeMotion()
3:   UpdateHistoryList()
4:   UpdateParametricCoordinates()
5:   if BoundaryConstraint() then
6:     ProjectConstraints()
7:     SolveSystem()
8:     UpdateParametricCoordinates()
9:     if BoundaryPuncture() then
10:      ReplaceTipConstraint()
11:      SolveSystem()
12:      UpdateHistoryListBoundary()
13:      UpdateParametricCoordinates()
14:   UpdateTransverseConstraints()
15:   SolveSystem()
```

thread is attached to the needle at its end, it will follow the needle's handle path as it drives through soft tissue. The needle and thread have the same topological representation but differ in their geometrical and mechanical representation. The needle is treated as a rigid body while a thread is treated as a soft body.

The intersection points, that are saved in the history list, serve as anchor points that bind the thread to the tissue and are used to relay displacement and forces, back and forth, between the thread and the embedding soft tissue, thus creating a coupling between the two representations that serves as a model for thread-tissue interaction.

Whenever the needle end passes an intersection point, the history cell is flagged and a different parametric update procedure is activated. This is due to that fact that needle sliding does not necessarily translate to an equal amount of thread sliding; the thread is non-rigid and has a different behavior. However, the parametric coordinate space is common to both needle and thread as they form a single topological model. As the thread moves through the soft tissue, the parametric coordinates of the intersection points are updated accordingly; the barycentric coordinates remain unchanged and are used to couple the thread to soft

tissue through algebraic constraints.

3.3.1 Pull Modeling

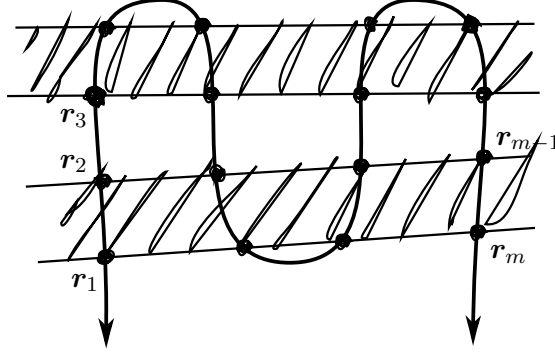


Figure 3.8 Sequence of puncture points in a running suture.

An example of a running suture is shown in (3.8) with entry and exit points numbered sequentially from \mathbf{r}_1 to \mathbf{r}_m . Under displacement control, the user is pulling on the pieces of the thread going through points \mathbf{r}_1 and \mathbf{r}_m . For simplicity of the presentation, we consider here the case of a frictionless interaction between the thread and tissue. The induced tissue deformation and internal stresses are due to the tensile forces in the thread and the contact forces from the closing boundaries.

The pull of the thread, when engaged, is modeled by an equality that constrains the length of the thread portion that lies outside the tissue:

$$\sum_{i=1}^{\frac{m}{2}-1} \left\| \mathbf{r}_{2i+1}^{[k+1]} - \mathbf{r}_{2i}^{[k+1]} \right\| = \sum_{i=1}^{\frac{n}{2}-1} \ell_i^{[k]} - \Delta \ell^{[k+1]}, \quad (3.32)$$

where $\mathbf{r}_i^{[k]}$ denotes the positions of tissue points at puncture locations at the k^{th} time step. The sum on the left-hand side in (3.32) is over the segments that join successive exit and entry points along the thread. Every term in the sum represents the length of such a segment as described below. $\ell_i^{[k]}$ is the length of the corresponding portion of the engaged straight thread at time step k and $\Delta \ell^{[k+1]}$ is the incremental pull enacted by the user at the $k+1$ time step through the puncture points \mathbf{r}_1 and \mathbf{r}_m . This constraint presumes, and is active only when, the

thread is fully engaged and has no slack. As long as the right-hand side is smaller than the actual length of the thread between the two end points, pulling causes pure sliding until the thread is fully engaged.

Another way of expressing this pull constraint that makes its complementarity clearer is:

$$\sum_{i=1}^{\frac{m}{2}-1} \left\| \mathbf{r}_{2i+1}^{[k+1]} - \mathbf{r}_{2i}^{[k+1]} \right\| + \left\| \mathbf{r}_1^{[k+1]} - \mathbf{r}_0 \right\| + \left\| \mathbf{r}_{m+1} - \mathbf{r}_m^{[k+1]} \right\| \leq \ell^{[k]}, \quad (3.33)$$

where \mathbf{r}_0 and \mathbf{r}_{m+1} are thread points at locations before first entry (\mathbf{r}_1) and after last exit (\mathbf{r}_m) of the suture. These points are under direct end-user positional control. $\ell^{[k]}$ is the actual thread length outside the tissue between points \mathbf{r}_0 and \mathbf{r}_{m+1} and can be approximated by using the thread-parametric coordinates at the puncture points to compute the length of the thread between the end points. When the left hand side of (3.33) is satisfied with strict inequality, the thread has slack and the corresponding Lagrange multiplier is zero. Only when it is satisfied with equality, does the thread get engaged and applies forces to deform the tissue and bring the sutured boundaries together.

As the suture closes additional contact constraints are introduced, as described in §2.2.6, to prevent the two boundaries from interpenetrating and hold them together as the suture tightens, and this could result in multiple collisions occurring during a single inter-frame. A straightforward way to handle these collisions is to include their associated constraints all at once in the system, however, this does not realistically model how the boundaries would interact as these collisions do not necessarily occur at the exact same time (only during the same inter-frame). A better way of handling multiple collisions is to order them using their associated timestamps, process the first collision in time, and disregard the rest. This technique results in less collisions being logged and more accurately models the interaction of the boundaries as they are brought together to close.

3.3.2 Distance Linearization

For efficient computations with the pull constraint of (3.33), the distance equations are linearized at step $k + 1$ by approximating the length of every thread line segment as the projection on its supporting direction from the previous step k . The justification for this linearization may be explained with reference to ⟨3.9⟩. Let \mathbf{r}_{2i} and \mathbf{r}_{2i+1} be two sequential puncture points in the volumetric mesh joining an outer portion of the thread. Pulling \mathbf{r}_{2i} and \mathbf{r}_{2i+1} close together should in principle decrease the length of the line segment, ℓ_i , joining the two punctures points together.

$$\ell_i = \|\mathbf{r}_{2i+1} - \mathbf{r}_{2i}\| = \sqrt{(\mathbf{r}_{2i+1} - \mathbf{r}_{2i})^\top (\mathbf{r}_{2i+1} - \mathbf{r}_{2i})}. \quad (3.34)$$

Let $\hat{\mathbf{d}}_i = (\mathbf{r}_{2i+1} - \mathbf{r}_{2i}) / \|\mathbf{r}_{2i+1} - \mathbf{r}_{2i}\|$ be the direction of the line segment joining \mathbf{r}_{2i} and \mathbf{r}_{2i+1} as illustrated in ⟨3.9⟩. At every step, ℓ_i can be computed using:

$$\begin{aligned} \|\mathbf{r}_{2i+1} - \mathbf{r}_{2i}\| &= \frac{1}{\|\mathbf{r}_{2i+1} - \mathbf{r}_{2i}\|} (\mathbf{r}_{2i+1} - \mathbf{r}_{2i})^\top (\mathbf{r}_{2i+1} - \mathbf{r}_{2i}) \\ &= (\mathbf{r}_{2i+1} - \mathbf{r}_{2i})^\top \frac{(\mathbf{r}_{2i+1} - \mathbf{r}_{2i})}{\|\mathbf{r}_{2i+1} - \mathbf{r}_{2i}\|} \\ &= (\mathbf{r}_{2i+1} - \mathbf{r}_{2i})^\top \hat{\mathbf{d}}_i, \end{aligned} \quad (3.35)$$

and is linearized by approximating the value of $\hat{\mathbf{d}}_i^{[k+1]}$ with $\hat{\mathbf{d}}_i^{[k]}$ when calculating the distance of a segment, resulting in:

$$\ell_i^{[k]} = \left\| \mathbf{r}_{2i+1}^{[k+1]} - \mathbf{r}_{2i}^{[k+1]} \right\| \approx \left(\mathbf{r}_{2i+1}^{[k+1]} - \mathbf{r}_{2i}^{[k+1]} \right)^\top \hat{\mathbf{d}}_i^{[k]}. \quad (3.36)$$

As the user pulls on the engaged thread, each pair of sequential suture points, \mathbf{r}_{2i} and \mathbf{r}_{2i+1} , is incrementally brought together by decreasing the distance that separates them using a constraint of the following form:

$$\left(\mathbf{r}_{2i+1}^{[k+1]} - \mathbf{r}_{2i}^{[k+1]} \right)^\top \hat{\mathbf{d}}_i^{[k]} = \kappa \left\| \mathbf{r}_{2i+1}^{[k]} - \mathbf{r}_{2i}^{[k]} \right\|; 0 < \kappa < 1. \quad (3.37)$$

As long as the end-user keeps pulling the thread, κ is decreased to bring the puncture points together. The amount by which κ is decreased can be approximated from the amount of pull exerted by the end-user at suture points \mathbf{r}_0 and \mathbf{r}_{m+1} . The

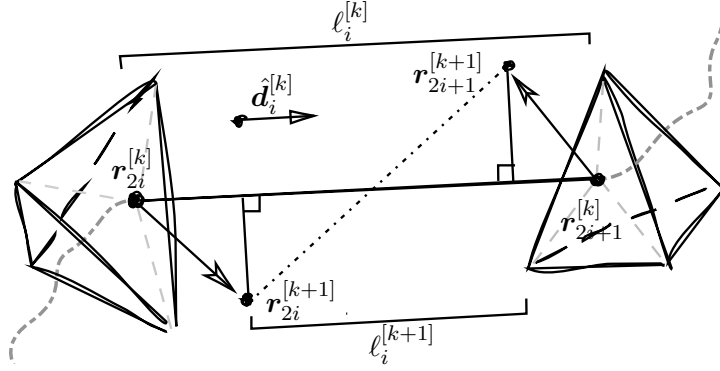


Figure 3.9 Distance approximation using linear projection for a single thread segment.

computation of $\hat{\mathbf{d}}_i$ is also suitably stabilized as the distance between \mathbf{r}_{2i} and \mathbf{r}_{2i+1} becomes small to avoid the near-zero denominator.

The visual and mechanical model of the thread can be rendered and animated using a deformable chain formulation as presented in [16, 32], for a more realistic thread-tissue visual interaction, where both boundary and internal points are used to impose the required constraints that guide the chain through the mesh. The scalar Lagrange multiplier associated with the pull constraint corresponds to the tensile force in the thread as it is being pulled which can be extrapolated, smoothed, and rendered to the end-user using the techniques described in [36] and [59].

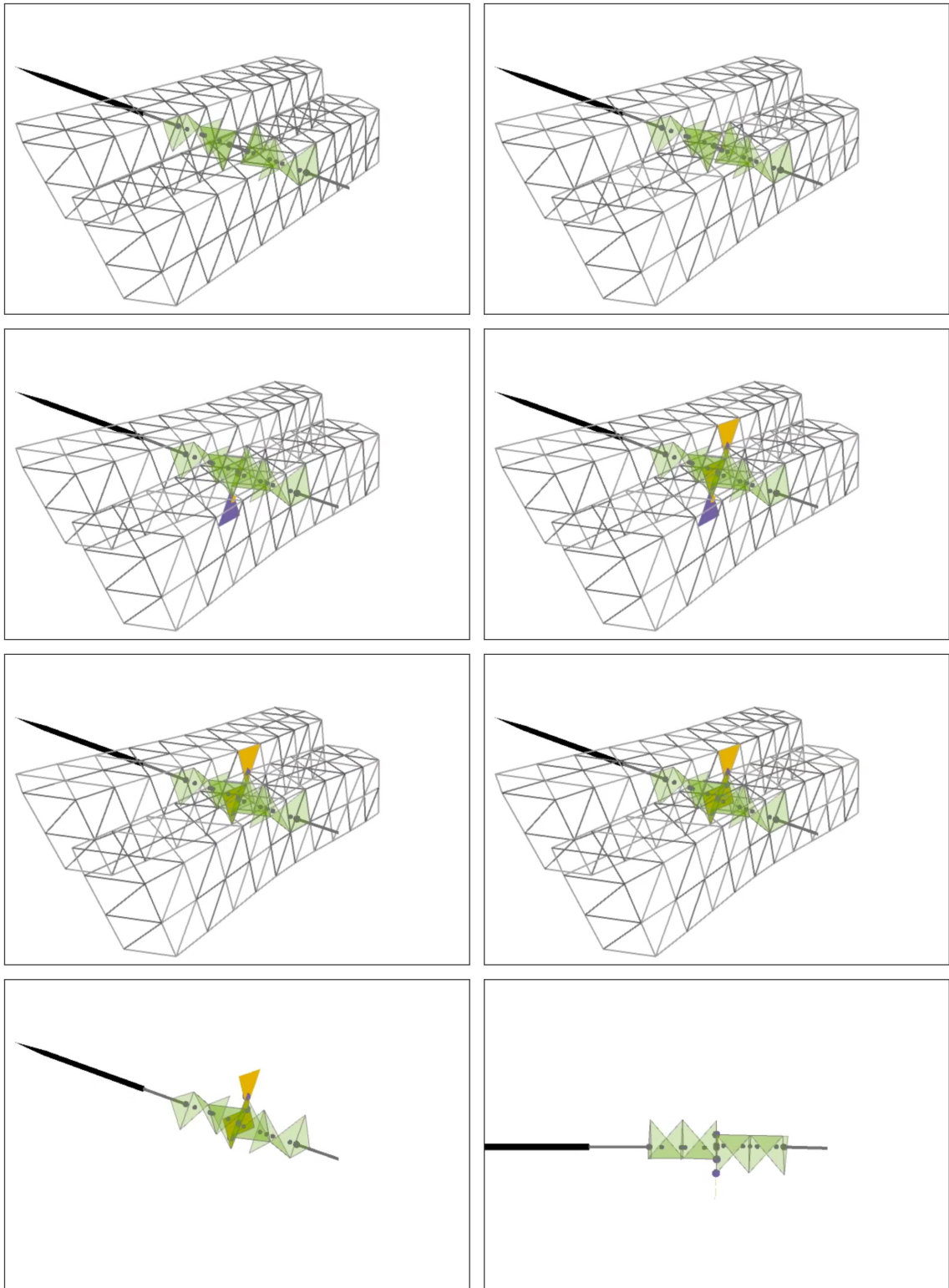


Figure 3.10 Tightening a suture using a sequence of thread pulls. The constraints holding the boundaries are dynamically being updated. The bottom left figure shows the list of faces that are used for modeling this interaction. The bottom right figure shows the resulting deformation after the suture is fully taut.

Chapter 4

Results

The presented techniques and methods have been integrated into a prototype suturing simulator through custom-designed modules for mesh processing and representation, visualization and rendering, finite element based simulation, and other techniques and utilities. Multiple off-the-shelf libraries are used namely for computational geometry routines, continuous deformable collision detection, and direct solution of sparse linear systems. In addition, multiple end-user interface devices were integrated to provide a rich end-user experience.

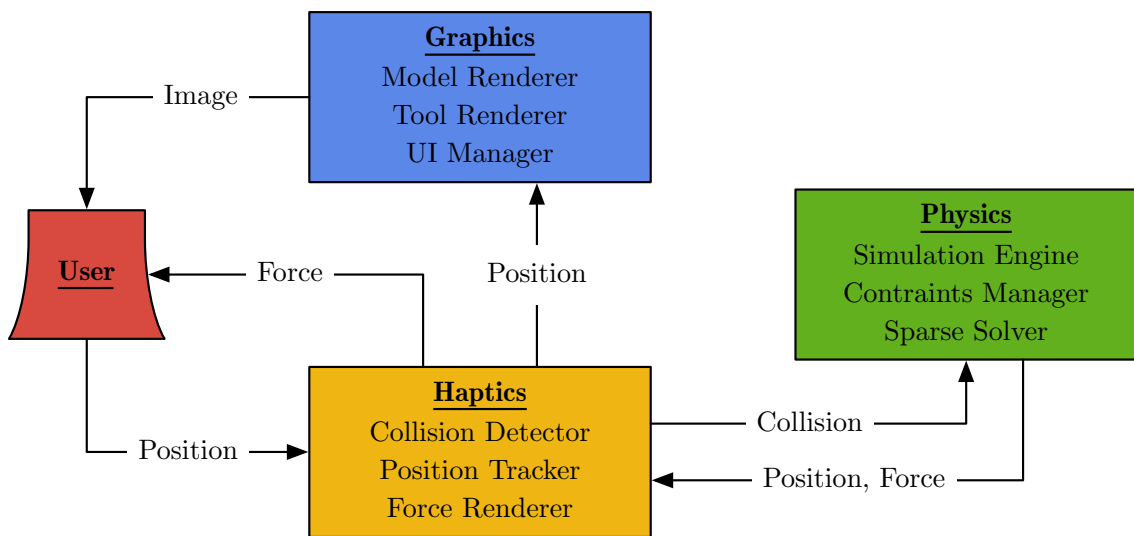


Figure 4.1 Overall system architecture.

The cross-platform implementation was developed using C++20 and OpenGL/GLSL 4.6, compiled using LLVM, GCC, and MSVC, and tested under macOS 11.5, Windows 11, and Fedora 34. The different components of the system are structured using a model-view-controller (MVC) architecture and separated into three major modules as depicted in the software architecture diagram in (4.1).

The suturing simulator prototype was deployed and tested on a bi-manual

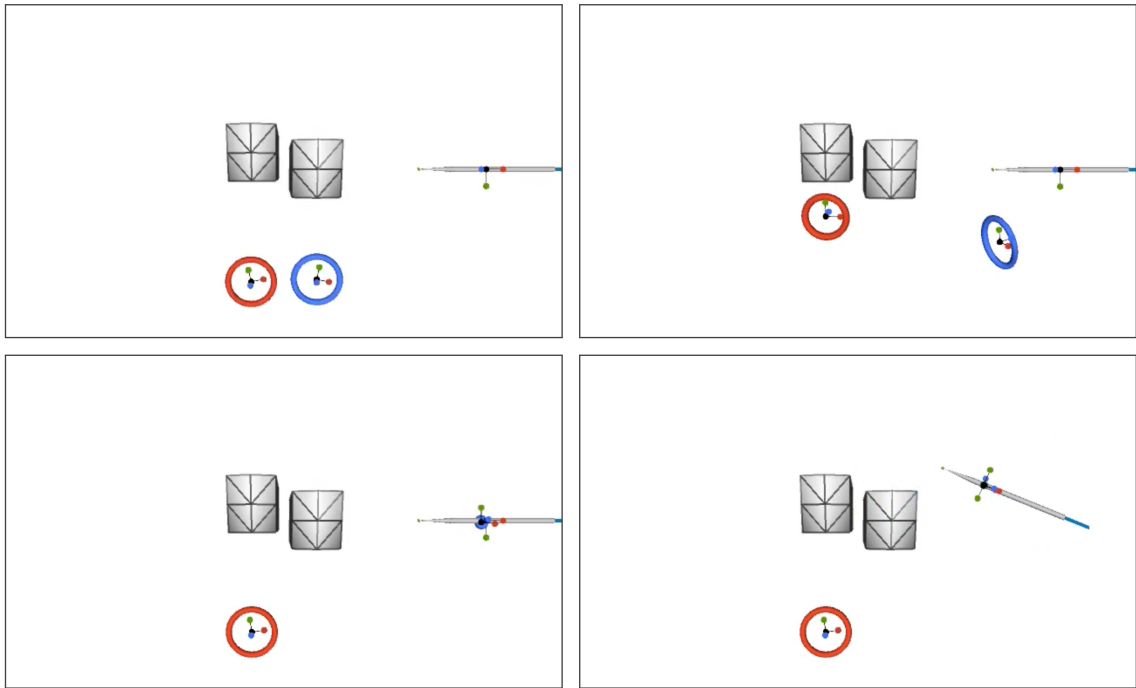


Figure 4.2 Needle motion relay. The end-user is controlling two 7-degrees of freedom (DOF) input devices and using them to grasp a needle by a handle and move along the domain. The supporting frames of the grasper and needle are interlocked and the motion is relayed directly from the input devices.

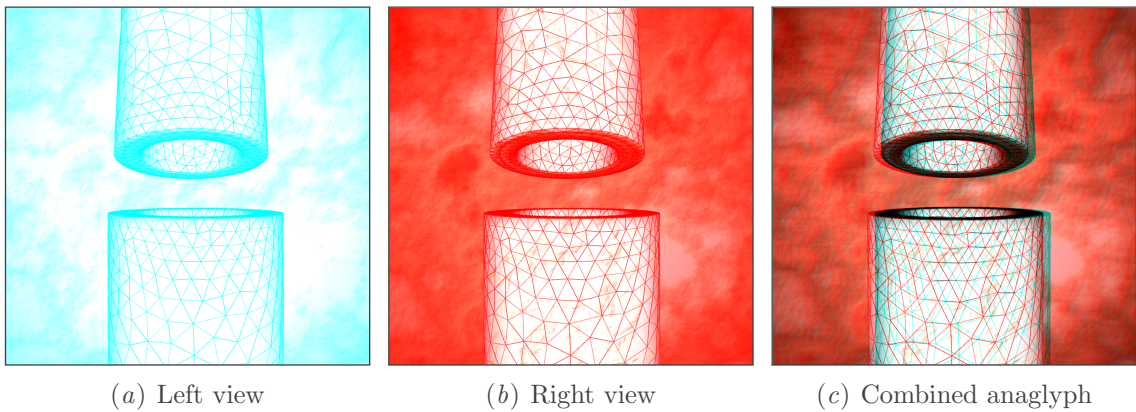


Figure 4.3 Stereo pair and red-cyan anaglyph using off-axis projection.



Figure 4.4 The bi-manual frame by Mimic Technologies used in their dV-Trainer simulator [66].

haptic console from Mimic Technologies (4.4). The console provides an immersive environment through an attached stereoscope, two spider masters that can recorder seven degrees of freedom (positions: $[x, y, z]$, angles: $[\alpha, \beta, \gamma]$, and grip opening: δ , that are used to control a virtual tool to grasp the needle as shown in (4.2), and five foot pedals for control various interface functions. The motors attached to the masters can render forces up to 15 N with a working space of $80 \times 40 \times 40$ cm. The console is designed to replicate the look and feel of the Da Vinci Xi console [65].

The simulator was tested on an HP Z8 workstation that connects to the bi-manual frame over a local network for input/output (I/O). The stereoscope is controlled separately through a direct connection to the graphics processing unit (GPU) and a technique involving off-axis frustums with parallel stereo pairs is used to render the stereoscopic scene as shown in (4.3).

Snapshots of an anastomosis procedure using our prototype simulator are

shown in ⟨4.5⟩, ⟨4.6⟩, and ⟨4.7⟩. ⟨4.5⟩ shows the initiation of the procedure with the needle puncturing (and deforming) the two vessels at four puncture points. In ⟨4.6⟩ and ⟨4.7⟩, running sutures are simulated using a varying number of punctures. Contact constraints prevent the two vessels from interpenetrating and allow the closure of the separating gap. Notice the high principal stress in the regions near the suture and its more uniform distribution when additional puncture points are used. Each step in these simulations takes an average of 30 ms to compute and render using a single 3.3 GHz core, resulting in an interactive refresh rate of 33 frames per second (FPS) on average during the course of the session.

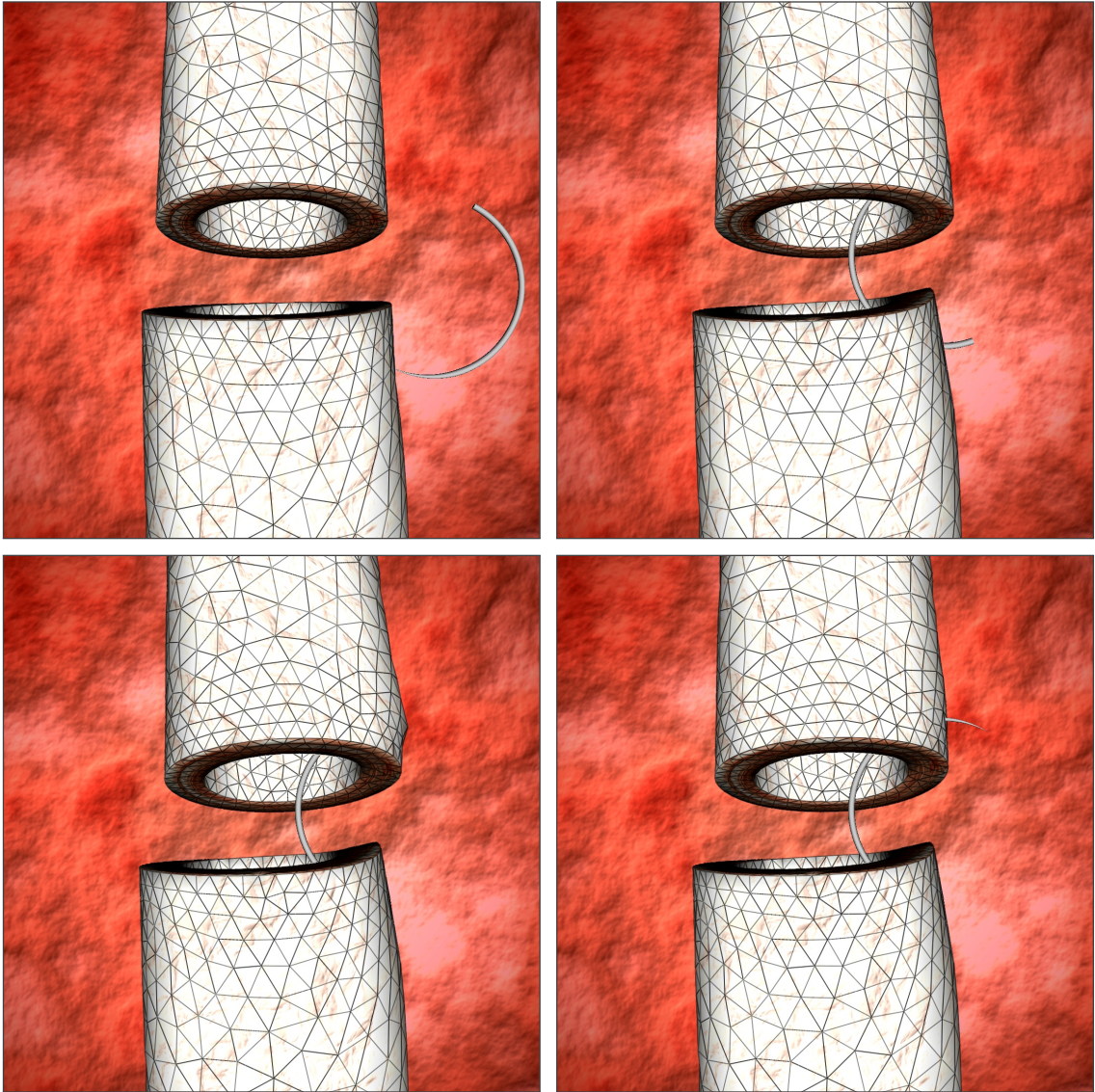


Figure 4.5 Needle driving using a $1/2$ -circle round body needle. The tubes have different diameters and are discretized with 1,650 nodes and 5,853 elements. The nodes of the top boundary of the upper tube and the bottom boundary of the lower tube are fixed.

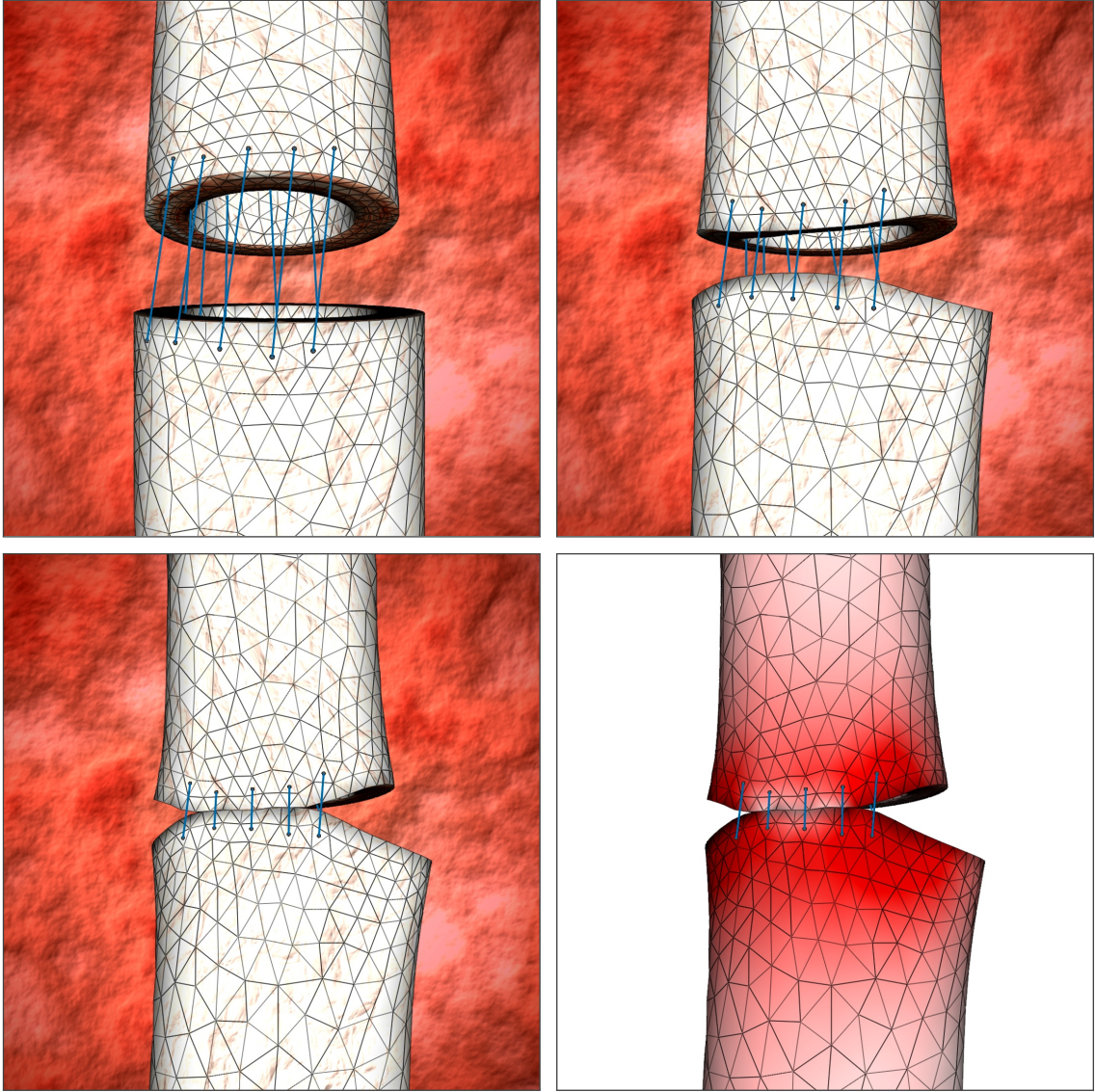


Figure 4.6 Closing of two boundaries using 20 puncture points. The tubes have different diameters and are discretized with 1,650 nodes and 5,853 elements. The principal stresses are visualized in the bottom-right figure.

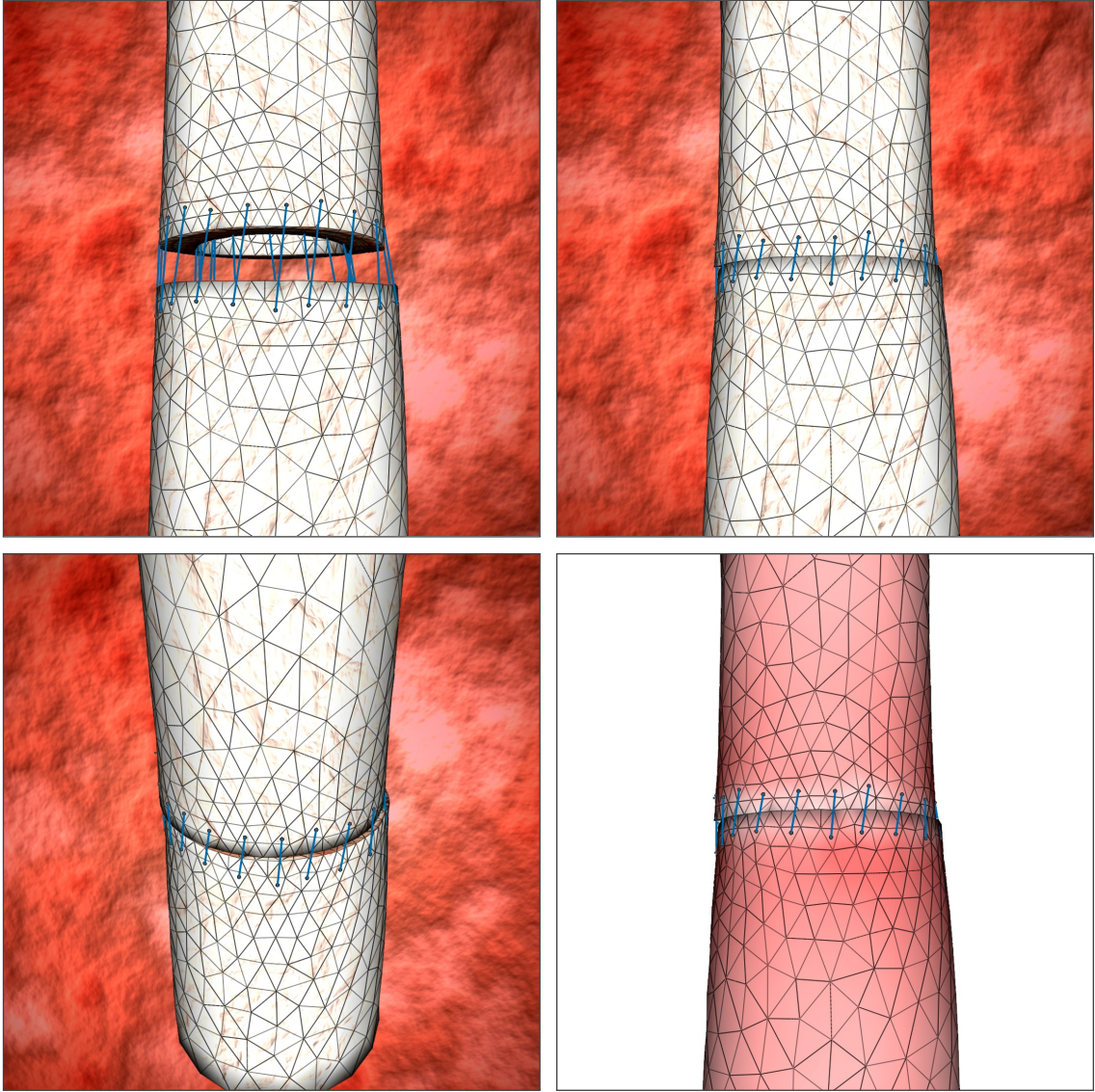


Figure 4.7 Closing of two boundaries using 50 puncture points. The tubes have different diameters and are discretized with 1,650 nodes and 5,853 elements. The principal stresses are visualized in the bottom-right figure.

Chapter 5

Conclusion

“It does not matter how slowly you go as long as you do not stop.”

—Confucius

5.1 Summary

This thesis presented effective techniques for needle driving and thread pulling that complement the large body of work on simulating sutures and can be used as building blocks for realizing virtual simulators.

Needles are represented as parametrized rigid bodies and their motion is decomposed into two disjoint components at every step through an energetic justification. This allows needle-tissue interaction to be treated in two disjoint and consecutive phases: a sliding phase that serves to track the position of the needle inside the tissue and neatly couples parametric needle points to barycentric tissue points, and a deforming (sticking) phase that uses this coupling to deform the tissue. Two most commonly used geometries are modeled under a single unifying representation: straight and semi-circular needles, and robust routines are used for detecting intersections between the needle and tissue as it slides through. Multiple edge cases are highlighted and dealt with for a robust tracking process.

Tissue is modeled as a soft body, discretized using 3-manifold meshes, and represented using a compact data structure that provides the necessary adjacency queries required for navigating and effectively tracking points through the tetrahedral mesh. This work complements the adjacency queries available through this representation by implementing two new queries: link of faces for a vertex and an edge, which are essential for tracking a point as it moves through the underlying

mesh. Soft models are simulated using a quasi-static co-rotational elastic finite element model and the underlying system of equations is solved using direct methods for sparse systems of linear equations.

Tissue-tissue interactions are modeled through the use of fast continuous deformable collision detection routines and enforced using complementarity constraints. This model of interaction generates forces that can be rendered to an end-user driving the system but does not handle fracture, tearing, or cutting effects and thus avoids the costly tasks of re-meshing the underlying domain and factorizing the underlying stiffness matrix at every iteration, all while providing plausible results for the various interactions. The complementarity constraints are also used to handle needle boundary punctures which serve to drive a thread through soft tissue. The thread is modeled as a linear chain and pulling effects are simulated through an approach that linearizes the distance between multiple boundary puncture points.

The presented techniques are integrated and tested in cross-platform prototype system, which allows for interactive simulation of suturing with high-resolution models through a simplified anastomosis scenario while making use of multiple human-computer interaction devices and immersive rendering techniques. These techniques form a robust basis for simple and efficient suturing simulation and are currently using as building blocks for a more realistic surgical simulator that can pass the test of face, content, and construct validity.

5.2 Future Work

Our work can be enhanced on multiple fronts. The needle-tissue interactions can be expanded to account for all kinds of interactions across the needle body and not just only those involving its tip. This kind of interaction, mostly involving the body of the needle deforming the soft tissue, can be expressed

using our complementarity constraints approach similarly to the way tissue-tissue interactions are modeled. The needle and thread can be modeled as deformable bodies and the tissue model can be extended to support discontinuous deformations with topological mesh changes. This extension allows the modeling of bending, friction, cutting, fracture, twisting, stretching, and other complex effects that occur when a needle and thread are being driven through soft tissue. It would also require a new approach for decomposing needle motion with minimal changes to the tracking and sticking phases. The physically-based model can also be upgraded to handle geometric and material non-linearities with dynamic relaxation to model velocity-dependent effects that affect the needle-tissue friction model. Knots can be simulated as well and integrated with the thread pulling model. This all has to be achieved while staying within the limits of the available computational budget and maintaining a good balance between the high-fidelity and real-time requirements as “there is no such thing as a free lunch.”

Parts of this thesis have been published in [47, 48, 50, 53] and were supported in part by Qatar Science & Technology Park and Qatar Foundation.

References

- [1] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. “Elastically Deformable Models”. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. Vol. 21. SIGGRAPH '87 4. New York, NY, USA: Association for Computing Machinery, 1987, pp. 205–214 (cited in p. 16).
- [2] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. “Dimension-Independent Modeling with Simplicial Complexes”. In: *ACM Transactions on Graphics* 12.1 (1993), pp. 56–102 (cited in p. 9).
- [3] Tomas Möller and Ben Trumbore. “Fast, Minimum Storage Ray-Triangle Intersection”. In: *Journal of Graphics Tools* 2.1 (1997), pp. 21–28. DOI: 10.1080/10867651.1997.10487468 (cited in p. 45).
- [4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Third Edition. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999 (cited in p. 26).
- [5] J. Brown, S. Sorkin, C. Bruyns, J.-C. Latombe, K. Montgomery, and M. Stephanides. “Real-Time Simulation of Deformable Objects: Tools and Application”. In: Seoul, South Korea. Seoul, South Korea: IEEE, 2001, pp. 228–258. DOI: 10.1109/ca.2001.982397 (cited in p. 33).
- [6] Jarek Rossignac, Alla Safonova, and Andrzej Szymczak. “3D Compression Made Simple: Edgebreaker on a Corner-Table”. In: *Shape Modeling International*. IEEE, 2001, pp. 278–283 (cited in p. 10).
- [7] S. DiMaio and S. Salcudean. “Needle Insertion Modelling for the Interactive Simulation of Percutaneous Procedures”. In: *Lecture Notes in Computer*

- Science* 2489 (2002). Ed. by Takeyoshi Dohi and Ron Kikinis. Proceedings of MICCAI, pp. 253–260 (cited in p. 30).
- [8] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. “Stable Real-Time Deformations”. In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’02. San Antonio, Texas: Association for Computing Machinery, 2002, pp. 49–54. DOI: 10.1145/545261.545269 (cited in pp. 22, 23).
- [9] Leila De Floriani and Annie Hui. “A Scalable Data Structure for Three-Dimensional Non-Manifold Objects”. In: *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. Sgp ’03. Aachen, Germany: Eurographics Association, 2003, pp. 72–82 (cited in pp. 9, 47).
- [10] S.P. DiMaio and S.E. Salcudean. “Needle Insertion Modeling and Simulation”. In: *IEEE Transactions on Robotics and Automation* 19.5 (2003), pp. 864–875. DOI: 10.1109/TRA.2003.817044 (cited in p. 31).
- [11] Matt LeDuc, Shahram Payandeh, and John Dill. “Toward Modeling of a Suturing Task”. In: *Proceedings of Graphics Interface*. 2003, pp. 273–279 (cited in p. 32).
- [12] M.D. O’Leary, C. Simone, T. Washio, K. Yoshinaka, and A.M. Okamura. “Robotic Needle Insertion: Effects of Friction and Needle Geometry”. In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE, 2003, pp. 1774–1780. DOI: 10.1109/robot.2003.1241851 (cited in p. 31).
- [13] Jeffrey Berkley, George Turkiyyah, Daniel Berg, Mark Ganter, and Suzanne Weghorst. “Real-Time Finite Element Modeling for Surgery Simulation: An Application to Virtual Suturing”. In: *IEEE Transactions on Visualization and Computer Graphics* 10.3 (2004), pp. 314–325 (cited in p. 30).

- [14] Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. USA: Cambridge University Press, 2004. DOI: 10.1017/CB09780511804441 (cited in p. 24).
- [15] Joel Brown, Jean-Claude Latombe, and Kevin Montgomery. “Real-Time Knot-tying Simulation”. In: *The Visual Computer* 20.2 (2004), pp. 165–179 (cited in p. 33).
- [16] Julien Lenoir, Philippe Meseure, Laurent Grisoni, and Christophe Chaillou. “A Suture Model for Surgical Simulation”. In: *Lecture Notes in Computer Science* 3078 (2004). Proceedings of ISMS, pp. 105–113 (cited in pp. 32, 61).
- [17] Matthias Müller and Markus Gross. “Interactive Virtual Materials”. In: *Proceedings of Graphics Interface 2004*. GI '04. London, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, pp. 239–246 (cited in p. 23).
- [18] Jason Cantarella, Michael Piatek, and Eric Rawdon. “Visualizing the Tightening of Knots”. In: *Proceedings of the IEEE Visualization*. VIS '05. IEEE Computer Society, 2005, pp. 575–582 (cited in p. 33).
- [19] Jessica R. Crouch, Chad M. Schneider, Josh Wainer, and Allison M. Okamura. “A Velocity-Dependent Model for Needle Insertion in Soft Tissue”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2005*. Ed. by James S. Duncan and Guido Gerig. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 624–632 (cited in p. 33).
- [20] S. DiMaio and S. Salcudean. “Needle Steering and Motion Planning in Soft Tissues”. In: *IEEE Transactions on Biomedical Engineering* 52.6 (2005), pp. 965–974 (cited in p. 30).
- [21] S. P. DiMaio and S. E. Salcudean. “Interactive Simulation of Needle Insertion Models”. In: 52 (2005), pp. 1167–1179. DOI: 10.1109/tbme.2005.847548 (cited in p. 30).

- [22] Orcun Goksel, Septimiu Salcudean, Simon DiMaio, Robert Rohling, and James Morris. “3D Needle-Tissue Interaction Simulation for Prostate Brachytherapy”. In: *Lecture Notes in Computer Science* 3749 (2005). Proceedings of MICCAI, pp. 827–834 (cited in p. 33).
- [23] Marcos Lage, Thomas Lewiner, Helio Lopes, and Luiz Velho. “CHF: A Scalable Topological Data Structure for Tetrahedral Meshes”. In: *Proceedings of the 18th Brazilian Symposium on Computer Graphics and Image Processing*. IEEE Computer Society, 2005, pp. 349–356 (cited in p. 10).
- [24] Marcos Lage, Thomas Lewiner, Helio Lopes, and Luiz Velho. “CHF: A Scalable Topological Data Structure for Tetrahedral Meshes”. In: *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*. Sibgrapi '05. IEEE Computer Society, 2005, pp. 349–356 (cited in p. 12).
- [25] Florent Nageotte, Philippe Zanne, Michel de Mathelin, and Christophe Doignon. “A Circular Needle Path Planning Method for Suturing in Laparoscopic Surgery”. In: *Proceedings of International Conference on Robotics and Automation*. IEEE, 2005, pp. 514–519 (cited in p. 32).
- [26] Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, Marie-Paule Cani, Francois Faure, Nadia Thalmann, Wolfgang Straßer, and Pascal Volino. “Collision Detection for Deformable Objects”. In: *Computer Graphics Forum* 24.1 (2005), pp. 61–81. DOI: 10.1111/j.1467-8659.2005.00829.x (cited in p. 31).
- [27] Fei Wang, Etienne Burdet, Vuillemin Ronald, and Hannes Bleuler. “Knot-tying with Visual and Force Feedback for VR Laparoscopic Training”. In: *Proceedings of 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2005 (cited in p. 33).

- [28] Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. Fundamentals of Algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2006 (cited in p. 26).
- [29] Naga K. Govindaraju, Ilknur Kabul, Ming C. Lin, and Dinesh Manocha. “Fast Continuous Collision Detection among Deformable Models using Graphics Processors”. In: *Computers & Graphics* 31.1 (2007), pp. 5–14. DOI: 10.1016/j.cag.2006.09.005 (cited in p. 31).
- [30] Mireille Grégoire and Elmar Schömer. “Interactive Simulation of One-dimensional Flexible Parts”. In: *Computer-Aided Design* 39.8 (2007), pp. 694–707 (cited in p. 33).
- [31] Lenka Jeřábková. “Interactive Cutting of Finite Elements based Deformable Objects in Virtual Environments”. PhD thesis. RWTH Aachen University, 2007 (cited in p. 21).
- [32] Blazej Kubiak, Nico Pietroni, Fabio Ganovelli, and Marco Fratarcangeli. “A Robust Method for Real-Time Thread Simulation”. In: *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*. VRST '07. Association for Computing Machinery, 2007, pp. 85–88 (cited in pp. 33, 61).
- [33] S. Laycock and A. Day. “A Survey of Haptic Rendering Techniques”. In: *Computer Graphics Forum* 56.1 (2007), pp. 50–65 (cited in p. 31).
- [34] Sarthak Misra, Kyle B. Reed, Andrew S. Douglas, K. T. Ramesh, and Allison M. Okamura. “Needle-Tissue Interaction Forces for Bevel-Tip Steerable Needles”. In: *2008 2nd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*. 2008, pp. 224–231. DOI: 10.1109/biorob.2008.4762872 (cited in p. 32).
- [35] Ajit Sachdeva, Carlos Pellegrini, and Kathleen Johnson. “Support for Simulation-Based Surgical Education Through American College of Surgeons”. In: *World Journal of Surgery* 32.2 (2008), pp. 196–207 (cited in p. 1).

- [36] Guillaume Saupin, Christian Duriez, and Stéphane Cotin. “Contact Model for Haptic Medical Simulation”. In: *Lecture Notes in Computer Science* 5104 (2008). Proceedings of ISBMS (cited in p. 61).
- [37] Ruediger Schmedding and Matthias Teschner. “Inversion Handling for Stable Deformable Modeling”. In: *The Visual Computer* 24.7 (2008), pp. 625–633 (cited in p. 23).
- [38] Nuttapon Chentanez, Ron Alterovitz, Daniel Ritchie, Lita Cho, Kris K. Hauser, Kenneth Y. Goldberg, Jonathan Richard Shewchuk, and James F. O’Brien. “Interactive Simulation of Surgical Needle Insertion and Steering”. In: *ACM Transactions on Graphics* 28.3 (2009), 88:1–10 (cited in p. 32).
- [39] Christian Duriez, Christophe Guébert, Maud Marchal, Stéphane Cotin, and Laurent Grisoni. “Interactive Simulation of Flexible Needle Insertions Based on Constraint Models”. In: *Lecture Notes in Computer Science* 5762 (2009). Proceedings of MICCAI 2009, pp. 291–299. DOI: 10.1007/978-3-642-04271-3 (cited in p. 31).
- [40] Carlos A. Felippa. *Advanced Finite Element Methods*. 2009 (cited in p. 20).
- [41] C. Guébert, C. Duriez, S. Cotin, J. Allard, and L. Grisoni. *Suturing Simulation Based on Complementarity Constraints*. Proceedings of SCA (Poster). 2009 (cited in p. 32).
- [42] Topraj Gurung and Jarek Rossignac. “SOT: Compact Representation for Tetrahedral Meshes”. In: *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*. SPM ’09. San Francisco, California: Association for Computing Machinery, 2009, pp. 79–88 (cited in p. 14).
- [43] Miguel A. Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. “Implicit Contact Handling for Deformable Objects”. In: *Computer Graphics*

- Forum* 28.2 (2009), pp. 559–568. DOI: 10.1111/j.1467-8659.2009.01396.x (cited in p. 31).
- [44] Shahram Payandeh and Fuhan Shi. “Interactive Multi-modal Suturing”. In: *Virtual Reality* 14.4 (2010), pp. 241–253. DOI: 10.1007/s10055-010-0174-6 (cited in p. 32).
- [45] Min Tang, Dinesh Manocha, and Ruofeng Tong. “Fast Continuous Collision Detection Using Deforming Non-Penetration Filters”. In: *Proceedings of I3D*. Association for Computing Machinery, 2010, pp. 7–13 (cited in pp. 27, 31).
- [46] Diana L. Diesen, Loretta Erhunmwunsee, Kyla M. Bennett, Kfir Ben-David, Basil Yurcisin, Eugene P. Ceppa, Philip A. Omotosho, Alexander Perez, and Aurora Pryor. “Effectiveness of Laparoscopic Computer Simulator Versus Usage of Box Trainer for Endoscopic Surgery Training of Novices”. In: *Journal of Surgical Education* 68.4 (2011), pp. 282–289. DOI: 10.1016/j.jsurg.2011.02.007 (cited in p. 2).
- [47] Georges Younes and Wajih Boukaram. “3D Immersive Surgery Simulation”. In: *Annual American University of Beirut Basic Biomedical Research Day*. 18. Beirut, Lebanon: American University of Beirut, 2011 (cited in p. 72).
- [48] Georges Younes, George Turkiyyah, and Julien Abinahed. “Demonstration Prototype of a High-Fidelity Robotic-Assisted Suturing Simulator”. In: *Qatar Foundation Annual Research Forum*. CSP16. Bloomsbury Qatar Foundation Journals, 2011, p. 270 (cited in p. 72).
- [49] Kup-Sze Choi, Sze-Ho Chan, and Wai-Man Pang. “Virtual Suturing Simulation Based on Commodity Physics Engine for Medical Learning”. In: *Journal of Medical Systems* 36.3 (2012), pp. 1781–1793. DOI: 10.1007/s10916-010-9638-1 (cited in p. 32).
- [50] Abdelaziz Farhat, Mohammed Alhaddad, Georges Younes, Tarek Elghazaly, Julien Abinahed, Abdulla Alansari, and George Turkiyyah. “New Evaluation

- Metrics Applied to Robotic Anastomosis”. In: *5th Hamlyn Symposium on Medical Robotics*. Imperial College London, 2012, pp. 91–92 (cited in p. 72).
- [51] Davis J.W. Lallas C.D. “Robotic Surgery Training with Commercially Available Simulation Systems in 2011: A Current Review and Practice Pattern Survey from The Society Of Urologic Robotic Surgeons”. In: *Journal of Endourology* 26.3 (2012), pp. 283–293 (cited in p. 2).
- [52] Dong Joon Lee and Doo Yong Lee. “A Method for Collision Detection between Needle and Tissues in the Needle Insertion Simulation”. In: *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*. 2013, pp. 113–118. DOI: 10.1109/iccas.2013.6703873 (cited in p. 31).
- [53] Georges Younes, Julien Abinahed, and George Turkiyyah. “Efficient Suturing of Deformable Models”. In: *Computational Biomechanics for Medicine: Models, Algorithms, and Implementation*. 7th Workshop on Computational Biomechanics for Medicine, 15th International Conference on Medical Image Computing and Computer Assisted Intervention. Springer Science+Business Media, 2013, pp. 75–84. DOI: 10.1007/978-1-4614-6351-1_8 (cited in p. 72).
- [54] Eugenia Yiannakopoulou, Nikolaos Nikiteas, Despina Perrea, and Christos Tsigris. “Virtual Reality Simulators and Training in Laparoscopic Surgery”. In: *International Journal of Surgery* 13 (2015), pp. 60–64. DOI: <https://doi.org/10.1016/j.ijvsu.2014.11.014> (cited in p. 2).
- [55] Andrea Moglia, Vincenzo Ferrari, Luca Morelli, Mauro Ferrari, Franco Mosca, and Alfred Cuschieri. “A Systematic Review of Virtual Reality Simulators for Robot-assisted Surgery”. In: *European Urology* 69.6 (2016), pp. 1065–1080. DOI: 10.1016/j.eururo.2015.09.021 (cited in p. 2).
- [56] Cianna Pender, Vladimir Kiselov, Qingzhao Yu, Jennifer Mooney, Patrick Greiffenstein, and John T. Paige. “All For Knots: Evaluating the

- Effectiveness of a Proficiency-Driven, Simulation-Based Knot Tying and Suturing Curriculum for Medical Students During Their Third-Year Surgery Clerkship”. In: *The American Journal of Surgery* 213.2 (2017), pp. 362–370. DOI: 10.1016/j.amjsurg.2016.06.028 (cited in p. 2).
- [57] Michael Pfandler, Marc Lazarovici, Philipp Stefan, Patrick Wucherer, and Matthias Weigl. “Virtual Reality-Based Simulators for Spine Surgery: A Systematic Review”. In: *The Spine Journal* 17.9 (2017), pp. 1352–1363. DOI: <https://doi.org/10.1016/j.spinee.2017.05.016> (cited in p. 2).
- [58] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. “Tetrahedral Meshing in the Wild”. In: *ACM Transactions on Graphics* 37.4 (2018), 60:1–60:14. DOI: 10.1145/3197517.3201353 (cited in p. 30).
- [59] Haiyang Ding, Hironori Mitake, and Shoichi Hasegawa. “Continuous Collision Detection for Virtual Proxy Haptic Rendering of Deformable Triangular Mesh Models”. In: *IEEE Transactions on Haptics* 12 (4 2019), pp. 624–634. DOI: 10.1109/toh.2019.2934104 (cited in p. 61).
- [60] Hugo Gonçalo Guedes, Zêmia Maria Câmara Costa Ferreira, Layra Ribeiro de Sousa Leão, Edna Frasson Souza Montero, José Pinhata Otoch, and Everson Luiz de Almeida Artifon. “Virtual Reality Simulator versus Box-Trainer to Teach Minimally Invasive Procedures: A Meta-Analysis”. In: *International Journal of Surgery* 61 (2019), pp. 60–68. DOI: 10.1016/j.ijisu.2018.12.001 (cited in p. 2).
- [61] Yixin Hu, Teseo Schneider, Xifeng Gao, Qingnan Zhou, Alec Jacobson, Denis Zorin, and Daniele Panozzo. “TriWild: Robust Triangulation with Curve Constraints”. In: *ACM Transactions on Graphics* 38.4 (2019). DOI: 10.1145/3306346.3323011 (cited in p. 30).

- [62] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. “Fast Tetrahedral Meshing in the Wild”. In: *ACM Transactions on Graphics* 39.4 (2020). DOI: 10.1145/3386569.3392385 (cited in p. 30).
- [63] Chi Jin, Liuyan Dai, and Tong Wang. “The Application of Virtual Reality in the Training Of Laparoscopic Surgery: A Systematic Review and Meta-Analysis”. In: *International Journal of Surgery* 87 (2021), p. 105859. DOI: 10.1016/j.ijjsu.2020.11.022 (cited in p. 2).
- [64] Jason D. Kelly, Timothy M. Kowalewski, Tim Brand, Anna French, Michael Nash, Lois Meryman, Nicholas Heller, Nancy Organ, Evalyn George, Roger Smith, Mathew D. Sorensen, Bryan Comstock, and Thomas S. Lendvay. “Virtual Reality Warm-Up Before Robot-Assisted Surgery: A Randomized Controlled Trial”. In: *Journal of Surgical Research* 264 (2021), pp. 107–116. DOI: 10.1016/j.jss.2021.01.037 (cited in p. 2).
- [65] Intuitive Surgical. *Da Vinci Surgical Systems*. 2021. URL: <https://www.intuitive.com/en-us/products-and-services/da-vinci/systems> (cited in p. 65).
- [66] Mimic Technologies. *dv-Trainer Simulator*. 2021. URL: <https://mimicsimulation.com/dv-trainer/> (cited in p. 65).