Friedrich Schiller University Jena
Faculty of Mathematics and Computer Science

Design and Implementation of High-Performance, Adaptive, and Robust Curve Smoothing on Surface Meshes and its Application to Medical Visualization

MASTER'S THESIS

for obtaining the academic degree

Master of Science (M.Sc.) in Mathematics

submitted by Markus Pawellek

born on May 7th, 1995 in Meiningen
Student Number: 144645

Primary Supervisor:	Kai Lawonn
Secondary Supervisor:	Noeska Smit

Bergen, February 22, 2023

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Zusammenfassung

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Acknowledgments

I am grateful to Kai Lawonn and Noeska Smit for their supervising, their helpful suggestions, and for our interesting discussions. Also great thanks to Ann Sommerfeld for assisting in writing the thesis and proofreading.

Contents

Contents i

List of Figures iii

List of Tables v

List of Definitions and Theorems vii

List of Code ix

List of Abbreviations and Acronyms xi

Symbol Table xiii

1 Introduction 1

2 Preliminaries 5
- 2.1 Differential Geometry of Curves . . . 5
- 2.2 Polyhedral Surfaces . . . 11

3 Related Work 15

4 Design 21
- 4.1 Overview of the Program Pipeline . . . 21
- 4.2 Polyhedral Surface Data Structure . . . 22
- 4.3 Surface Mesh Preprocessing . . . 23
- 4.4 Discrete Surface Curve Data Structure . . . 24
- 4.5 Initial Curve Selection . . . 24
- 4.6 Unfolding . . . 25
- 4.7 Unfolding with Desired Curvature . . . 26
- 4.8 Desired Curvature Stencil . . . 27
- 4.9 Desired Curvature Mapping . . . 27
- 4.10 Curve Smoothing Algorithm . . . 27

5 Implementation 29
- 5.1 Serial Implementation on CPU . . . 29
- 5.2 Parallelization and Vectorization . . . 29
- 5.3 Parallel Implementation on the CPU . . . 29
- 5.4 Implementation on the GPU . . . 29

6 Testing 31
- 6.1 Two-Dimensional Case . . . 31
- 6.2 Three-Dimensional Case . . . 31

7 Application 33

8 Evaluation and Results 35

9 Conclusions and Future Work **37**

References **39**

A Analysis on Manifolds **i**

B Mathematical Proofs **v**

C Further Code **vii**

List of Figures

2.1 Examples of Smooth Parameterized Curves 6
3.2 Examples of Discrete Geodesics 15
3.3 State-of-the-Art Discrete Geodesic Tracing 16
3.4 Examples of Subdivision Curves by Morera, Velho, and Carvalho (2008) . . . 17
3.5 Examples of Snakes by Lee and Lee (2002) 17
3.6 Examples of Energy-Minimizing Splines by Hofer and Pottmann (2004) . . . 18
3.7 Examples of Bézier Splines by Mancinelli et al. (2022) 19
3.8 Examples of Curve Smoothing by Lawonn et al. (2014) 20
4.9 Program Pipeline Stages 21
4.10 Polyhedral Surface Data Structure 23
4.11 Examples of Polyhedral Surfaces 23
4.12 2D Unfolding Primitive 25
4.13 Unfolding of two Triangles 25

List of Tables

List of Definitions and Theorems

2.1 Definition: Open and Closed Parameterized Curves . . . 5
2.2 Definition: Regular and Simple Parameterized Curves . . . 7
2.1 Corollary: Simple, Regular Curves are Embeddings . . . 7
2.3 Definition: Length of Curves . . . 7
2.2 Lemma: Regular curves can be parameterized by arc length . . . 8
2.4 Definition: Canonical Parameterization by Arc Length . . . 9
2.5 Definition: Curvature of Regular Curves . . . 9
2.6 Definition: Geodesic and Normal Curvature of Curves . . . 9
2.3 Corollary: Relationship of Geodesic and Normal Curvature . . . 10
2.7 Definition: Oriented Curvatures of Curves . . . 10
2.8 Definition: Geodesic . . . 10
2.9 Definition: Triangle . . . 11
2.10 Definition: Polyhedral Surface and Surface Mesh . . . 12
2.11 Definition: Oriented Surface Mesh . . . 12
2.12 Definition: Discrete Surface Mesh Curve . . . 12
2.13 Definition: Surface Mesh Curve without Artifacts . . . 13
2.14 Definition: Total Vertex Angle . . . 13
2.15 Definition: Curve Angle . . . 13
2.16 Definition: Discrete Geodesic Curvature . . . 14
2.17 Definition: Discrete Geodesic Curvature for Boundaries . . . 14
2.18 Definition: Discrete Geodesic Problems . . . 14
2.4 Lemma: Straightest and shortest geodesics are surface mesh curves without artifacts. . . . 14
4.1 Lemma: Unfolding leads to geodesic . . . 26
A.1 Definition: Topological Manifold . . . i
A.2 Definition: Smooth Manifold . . . i
A.3 Definition: Smooth Maps . . . i
A.4 Definition: Tangential Space . . . ii
A.5 Definition: Differential . . . ii
A.6 Definition: Smooth Embedding . . . ii
A.7 Definition: Embedded Submanifold . . . ii
A.1 Theorem: The image of an embedding is an embedded submanifold . . . ii
A.8 Definition: Vector Bundle . . . ii
A.9 Definition: Tensorbundle . . . iii
A.10 Definition: Riemannian Manifold . . . iii
A.2 Theorem: For every smooth manifold, there exists a Riemannian metric . . . iii
A.11 Definition: Normal Space . . . iii

List of Code

4.1 Polyhedral Surface 22

List of Abbreviations and Acronyms

Abbreviation	Definition
iid	Independently and Identically Distributed
CDF	Cumulative Distribution Function
SLLN	Strong Law of Large Numbers
LTE	Light Transport Equation
API	Application Programming Interface
RAII	Resource Acquisition is Initialization
SFINAE	Specialization Failure is not an Error
STL	Standard Template Library

Symbol Table

Symbol	Definition
Logic	
$\exists \ldots : \ldots$	There exists . . ., such that . . .
$a \coloneqq b$	a is defined by b.
Set Theory	
$\{\ldots\}$	Set Definition
$\{\ldots \mid \ldots\}$	Set Definition with Condition
$x \in A$	x is an element of the set A.
$A \subset B$	The set A is a subset of the set B.
$A \cap B$	Intersection — $\{x \mid x \in A \text{ and } x \in B\}$ for sets A, B
$A \cup B$	Union — $\{x \mid x \in A \text{ or } x \in B\}$ for sets A, B
$A \setminus B$	Relative Complement — $\{x \in A \mid x \notin B\}$ for sets A, B
$A \times B$	Cartesian Product — $\{(x, y) \mid x \in A, y \in B\}$ for sets A and B
A^n	n-fold Cartesian Product of Set A
$\emptyset$	Empty set — $\{\}$.
$\#A$	Number of Elements in the Set A
$\mathcal{P}(A)$	Power Set of Set A
Special Sets	
$\mathbb{N}$	Set of Natural Numbers
$\mathbb{N}_0$	$\mathbb{N} \cup \{0\}$
$\mathbb{P}$	Set of Prime Numbers
$\mathbb{Z}$	Set of Integers
$\mathbb{Z}_n$	Set of Integers Modulo n
$\mathbb{F}_m$	Finite Field with $m \in \mathbb{P}$ Elements
$\mathbb{F}_m^{p \times q}$	Set of $p \times q$-Matrices over Finite Field $\mathbb{F}_m$
$\mathbb{F}_2$	Finite Field of Bits
$\mathbb{F}_2^n$	Set of n-bit Words
$\mathbb{R}$	Set of Real Numbers
$\mathbb{R}^n$	Set of n-dimensional Real Vectors
$\mathbb{S}^2$	Set of Directions — $\left\{x \in \mathbb{R}^3 \mid \|x\| = 1\right\}$
Functions	
$f \colon X \to Y$	f is a function with domain X and range Y.
id_X	Identity Function over the Set X
$f \circ g$	Composition of Functions f and g
f^{-1}	Inverse Image of Function f
f^n	n-fold Composition of Function f
Bit Arithmetic	
$x_{n-1} \ldots x_1 x_0$	n-bit Word x of Set $\mathbb{F}_2^n$
$x \leftarrow a$	Left Shift of all Bits in x by a
$x \rightarrow a$	Right Shift of all Bits in x by a
$x \circlearrowleft a$	Circular Left Shift of all Bits in x by a
$x \oplus y$	Bit-Wise Addition of x and y
$x \odot y$	Bit-Wise Multiplication of x and y
$x \mid y$	Bit-Wise Or of x and y

Symbol	Definition
Probability Theory	
$\mathcal{B}(\mathbb{R})$	Borel σ-Algebra over $\mathbb{R}$
$(\Sigma, \mathcal{A})$	Measurable Space over Σ with σ-Algebra $\mathcal{A}$
λ	Lebesgue Measure
$\int_U f \,\mathrm{d}\lambda$	Lebesgue Integral of f over U
$\mathrm{L}^2(U, \lambda)$	Set of Square-Integrable Functions over the Set U with Respect to the Lebesgue Measure λ
$(\Omega, \mathcal{F}, P)$	Probability Space over Ω with σ-Algebra $\mathcal{A}$ and Probability Measure P
$\int_\Omega X \,\mathrm{d}P$	Integral of Random Variable X with respect to Probability Space $(\Omega, \mathcal{A}, P)$
$\int_\Omega X(\omega) \,\mathrm{d}P(\omega)$	$\int_\Omega X \,\mathrm{d}P$
P_X	Distribution of Random Variable X
$\mathbb{E}\, X$	Expectation Value of Random Variable X
$\mathrm{var}\, X$	Variance of Random Variable X
$\sigma(X)$	Standard Deviation of Random Variable X
$\mathbb{1}_A$	Characteristic Function of Set A
δ_ω	Dirac Delta Distribution over $\mathbb{S}^2$ with respect to $\omega \in \mathbb{S}^2$
$\bigotimes_{n \in I} P_n$	Product Measure of Measures P_n Indexed by the Set I
Miscellaneous	
$(x_n)_{n \in I}$	Sequence of Values x_n with Index Set I
$\lvert x \rvert$	Absolute Value of x
$\lVert x \rVert$	Norm of Vector x
$x \mod y$	x Modulo y
$\gcd(\rho, k)$	Greatest Common Divisor of ρ and k
$\max(x, y)$	Maximum of x and y
$\lim_{n \to \infty} x_n$	Limit of Sequence $(x_n)_{n \in \mathbb{N}}$
$\sum_{k=1}^{n} x_k$	Sum over Values x_k for $k \in \mathbb{N}$ with $k \leq n$
$\dim X$	Dimension of X
$\lceil x \rceil$	Ceiling Function
$\langle x \mid y \rangle$	Scalar Product
$[a, b]$	$\{x \in \mathbb{R} \mid a \leq x \leq b\}$
(a, b)	$\{x \in \mathbb{R} \mid a < x < b\}$
$[a, b)$	$\{x \in \mathbb{R} \mid a \leq x < b\}$
Constants	
∞	Infinity
π	$3.1415926535\ldots$ — Pi
Units	
$1\,\mathrm{B}$	$1\,\mathrm{Byte} = 8\,\mathrm{bit}$
$1\,\mathrm{GiB}$	$2^{30}\,\mathrm{B}$
$1\,\mathrm{s}$	$1\,\mathrm{Seconds}$
$1\,\mathrm{min}$	$1\,\mathrm{Minutes} = 60\,\mathrm{s}$
$1\,\mathrm{GHz}$	$1\,\mathrm{Gigahertz} = 10^9\,\mathrm{Hertz}$

1 Introduction

Nowadays, the majority of application domains vital to the life of humanity is supported by computer-aided systems. These are typically programs that provide a set of tools to facilitate the automatic generation, transfer, manipulation, and visualization of domain-specific data by keeping user interaction at a required minimum. Computer systems have enabled humanity to streamline processes and to abstract and encapsulate low-level tasks. As a consequence, this resulted in the ability to solve harder problems even more efficiently.

Especially in the area of medicine, examples such as the resection of liver tumors for long-term survival (Alirr and Abd. Rahni 2019) and osteotomy planning (Zachow et al. 2003), that involves reshaping and realigning bones to repair or fix bone-specific issues, show that the use of computer-aided systems for surgery planning reduces the duration of treatment and heavily increases the chance of long-term survival. Both of the named medical applications use curves on the two-dimensional reconstructed surface of scanned medical objects, such as livers and bones, to represent and visualize surgery cuts. The reconstructed surfaces will thereby be provided as triangular meshes and are often referred to as surface meshes.
(Alirr and Abd. Rahni 2019; Zachow et al. 2003)[1]

By construction, initially chosen curves on these surfaces are jagged due to the finite precision of the underlying mesh and emit curvature noise that is not neglectable and perceivable by the human eye. Hence, a smoothing process is applied to initial curves to reduce their overall curvature and attain surface cuts with well-defined properties. In general, the result of curve smoothing might strongly deviate from the initially given curve to fulfill the given constraints. For medical surface cutting applications, though, the shape of an initial curve is defined by domain experts, such as physicians or bioengineers, and most likely indicates relevant anatomical landmarks or surface regions. Thus, under these circumstances, the smoothing additionally requires the resulting curve to be close to its original such that no essential information is lost during the process. (Lawonn et al. 2014)

Futhermore, it is a matter of fact, that curves on surface meshes and algorithms for smoothing them are basic building blocks for mesh processing and segmentation (Ji et al. 2006; Kaplansky and Tal 2009). Consequently, their fundamental role in the areas of computer-aided geometric design, computer graphics, and visualization, that are heavily based on mesh processing, is unconcealable. So, curve smoothing on surface meshes is not only relevant in specific areas of medicine but is a generally applicable and important tool to many other domains of applications building on the above research areas. Further domain areas, such as machine learning (Benhabiles et al. 2011; Park et al. 2019) and engineering, therefore provide many more direct and significant applications.

Besides their mathematical correctness and convergence, curve smoothing algorithms should exhibit a certain level of adaptivity with respect to the given surface mesh and its initially chosen curve. Surface meshes are most typically an irregular grid of triangular faces that may highly vary in diameter and area. In addition, the initial curve might be extreme concerning its length, curvature, and overall shape. An algorithm to smooth curves on surface meshes needs to adapt to all these situations and still figure out the best possible result that abides to the given criteria. In conjunction with its correctness, this also means that such an algorithm needs to be robust for many different kinds of scenarios, such as self-intersecting curves and noisy surface geometries, that may result in wrong calculations based on the finite

[1] In this thesis, citations concerning a whole paragraph will be given after the last sentence of the very paragraph.

precision of floating-point values. Yet another property to take into account is the efficiency of the algorithm. To seamlessly integrate curve smoothing into the user interface of a computer-assisted system for domain-specific applications, it at least needs to provide an interactive up to real-time performance. (Lawonn et al. 2014)

There are a few already existing algorithms for producing smoothed curves on surface meshes (Hofer and Pottmann 2004; Lawonn et al. 2014; Mancinelli et al. 2022; Martínez, Carvalho, and Velho 2007). Still, the implementation and API design of such algorithms is assumed to be an involved task and error-prone when the programmer intends to apply the algorithm on a wide variety of cases. All the given references define their algorithm and explain its properties in great detail. They compare the quality of generated curves to alternative algorithms and describe the algorithm's programming procedures at least with respect to a high-level point of view based on pseudocode. However, the very low-level details about the composition of data structures, advice for an implementation in a specific programming language, or ways to handle difficult corner cases are left out. This makes the comparison of the performance and robustness of algorithms much harder and unreproducible, because custom implementations would need to be used. Furthermore, up to this point there is no widely accepted metric to compare the smoothness of two different generated curves which leads to highly subjective treatment and evaluation of different algorithms.

For the design and implementation of a basic framework for curve smoothing on surfaces that allows for high-performance, reproducibility, and robustness, adequate candidates are the modern standards of the C++ programming language in conjunction with the OpenGL graphics API. C++ is a multi-paradigm language that integrates many different programming styles, such as object-oriented, functional, and data-oriented programming. It is still the de-facto standard for graphics applications and well-known to be one of the fastest languages in the world which incorporates low-level programming based on assembler routines and efficient high-level abstraction mechanisms, like template meta programming. The design of the whole language keeps on advancing to make programs faster and easier to develop. In the most common cases, C++ can be seen as a superset of the older C programming language which is typically used by other programming languages to provide the possibility of code being called from different languages. Therefore the users of the framework are not even restricted to use C++ but instead are able to use other languages, like Python, to communicate with a C interface to achieve similar results. OpenGL is the open-source graphics API that allows programs to efficiently communicate and interact with the driver of the graphics card to visualize provided data independently of the manufacturer or the operating system. By using them, no strong constraints are imposed on the software environment that the software framework is running on. Both tools allow for a sophisticated modularization of the whole framework. So, no user needs to pay for features that are not needed.
(*cppreference.com* n.d.; Meyers 2014; *OpenGL: The Industry's Foundation for High Performance Graphics* 2023; Reddy 2011; *Standard C++ Foundation* 2023; Stroustrup 2014; Vandevoorde, Josuttis, and Gregor 2018)

In this thesis, precisely in sections 4 and 5, we develop a new library and program, called *reflex*[2], using the C++ programming language in conjunction with OpenGL graphics API. *reflex* implements parallelized and tweaked variants of the curve smoothing algorithm given by Lawonn et al. (2014) on the CPU and GPU which should be applicable in a wide variety

[2]Markus Pawellek (2023). *reflex. Reactive and Flexible Curve Smoothing on Surface Meshes*. URL: https://github.com/lyrahgames/reflex (visited on 01/15/2023).

of cases. Hereby, a special emphasis lies on the robust and fast implementation for medical purposes. The program and library are open-source and can be found on GitHub. The necessary theoretical background to understand the design- and the implementation-specific aspects is given in the section 2. Here, we will give a brief introduction to differential geometry, polyhedral manifolds, and computer architecture. A mathematical rigorous discussion about the algorithm will be part of section 4 to properly encapsulate all the information specific to the implementations. Section 3 refers to the previous work concerning general curves, geodesics and the smoothing of curves on surfaces. At the end in section 7, we apply the constructed algorithm to the problem of segmentation of lung lobes (Park et al. 2019). In the sections 8 and 9, the evaluation is shown followed by a discussion dealing with further improvements.

2 Preliminaries

To systematically approach the design and implementation of curve smoothing algorithms for surface meshes, basic knowledge in the topic of differential geometry of curves and its application and generalization to polyhedral surfaces and surface mesh curves is administrable. It allows to rigorously formulate the initial and boundary value problems for discrete geodesics which build the foundation of curve smoothing as geodesics can be interpreted as the class of smoothest curves. Alas, differential geometry heavily builds on the mathematical tools given in the topic of analysis on manifolds and, as a consequence, the reader is assumed to be familiar with its basic concepts. For convenience, section A in the appendix still provides a brief introduction to the topic to remind of and clarify notations.

2.1 Differential Geometry of Curves

The theory of smooth curves in space or smooth surfaces is one of the main concerns of classical differential geometry. Therefore, we refer to some standard textbooks, namely Goldhorn, Heinz, and Kraus (2009), Carmo (2016), Kühnel (2013), and Stahl and Stenson (2013), for a more excessive and thorough introduction on this topic. For the purpose of consistent notation and as a reminder, the main concepts of smooth curves are briefly given in the following in the sense of classical and modern differential geometry based on these books.

In the fields of analysis and numerical mathematics, it is a natural approach to extend a well-founded and -understood theory for smooth objects to their discrete counterparts and vice versa. Hereby, discrete or smooth objects will be typically characterized by the limit of sequences of smooth or discrete objects, respectively. This procedure may then allow for a generalization of properties and statements from one to the other case. Surface mesh curves, as they will be defined in section 2.2, can exactly be seen as such discrete counterparts to the shapes of special classes of smooth parameterized curves (Polthier and Schmies 2006).
(Cheney and Kincaid 2008; Elstrodt 2018; Forster 2016; Goldhorn, Heinz, and Kraus 2009)

DEFINITION 2.1: Open and Closed Parameterized Curves

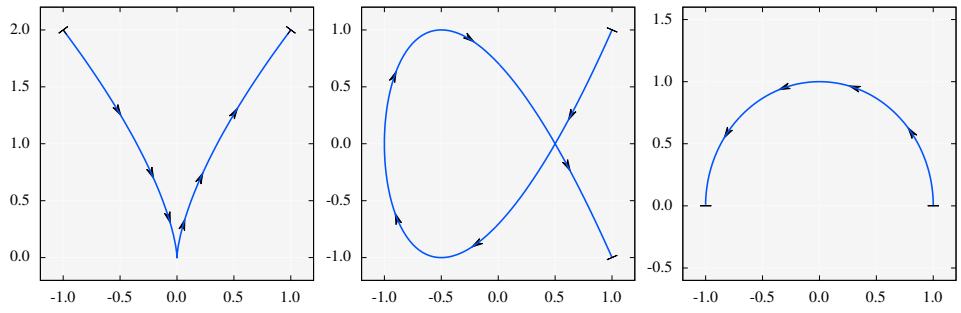
Let $n \in \mathbb{N}$, $k \in \mathbb{N}_\infty$, and $[a,b] \subset \mathbb{R}$ be a compact interval.

An (open) (n-dimensional) (parameterized) curve (of class C^k) is a k-times continuously differentiable function $\gamma\colon [a,b] \to \mathbb{R}^n$ on the compact interval $[a,b]$.

A closed (n-dimensional) (parameterized) curve (of class C^k) is a k-times continuously differentiable function $\gamma\colon \mathbb{T}^1[a,b] \to \mathbb{R}^n$ on the respective one-dimensional topological torus $\mathbb{T}^1[a,b]$.

Parameterized curves of class C^∞ are also called smooth.

In the sense of this definition, a curve needs to at least provide some kind of smoothness in the form of continuous differentiability. Trying only to rely on a continuity property, the definition would also include pathological cases, such as space-filling curves that cover a whole square in the plane (Kühnel 2013). Other generalizations therefore rely on the use of either rectifiable or piecewise continuously differentiable curves.



(a) *Semicubical Parabola*

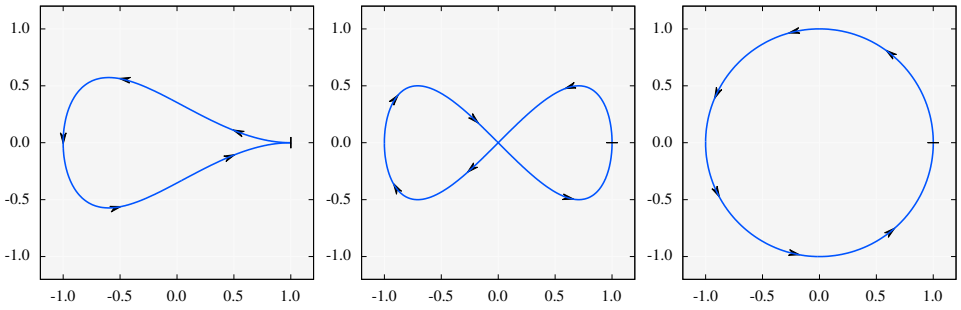
$$\gamma\colon [-1,1] \to \mathbb{R}^2$$
$$\gamma(t) \coloneqq \begin{pmatrix} t^3 \\ 2t^2 \end{pmatrix}$$

(b) *Lissajous Curve*

$$\gamma\colon [0,2\pi] \to \mathbb{R}^2$$
$$\gamma(t) \coloneqq \begin{pmatrix} \cos t \\ \cos\left(\frac{3}{2}t\right) \end{pmatrix}$$

(c) *Half-Circle*

$$\gamma\colon [0,\pi] \to \mathbb{R}^2$$
$$\gamma(t) \coloneqq \begin{pmatrix} \cos t \\ \sin t \end{pmatrix}$$



(d) *Teardrop Curve*

$$\gamma\colon \mathbb{T}^1[0,2\pi] \to \mathbb{R}^2$$
$$\gamma(t) \coloneqq \begin{pmatrix} \cos(t) \\ \sin(t)\sin^3\left(\frac{1}{2}t\right) \end{pmatrix}$$

(e) *Lissajous Curve*

$$\gamma\colon \mathbb{T}^1[0,2\pi] \to \mathbb{R}^2$$
$$\gamma(t) \coloneqq \begin{pmatrix} \cos t \\ \frac{1}{2}\sin 2t \end{pmatrix}$$

(f) *Circle*

$$\gamma\colon \mathbb{T}^1[0,2\pi] \to \mathbb{R}^2$$
$$\gamma(t) \coloneqq \begin{pmatrix} \cos t \\ \sin t \end{pmatrix}$$

Figure 2.1: ***Examples of Smooth Parameterized Curves***

All the plots show a smooth two-dimensional parameterized curve γ of a different class. The first row from left to right shows a general curve, a regular curve with self-intersection, and a regular and simple curve parameterized by arc length. The second row shows closed counterparts, respectively. Hereby, the black arrows indicate the orientation of the chosen parameterization and the black lines its start and end.

Defining closed curves to be smooth functions on the topological space of a one-dimensional torus is a technicality that automatically takes care of the agreement of values and derivatives at the two boundary points which are glued together (Carmo 2016; Stahl and Stenson 2013). The change of topological requirements for a closed curve still allows to interpret it as a general curve. For this reason, when referring to parameterized curves in the following, closed and open curves are meant. To get an impression of the implications of this definition, in figure 2.1 typical examples of smooth curves are shown.

For most of the considered applications, the actual parameterization of a curve is not essential. Only the shape, given by its image, is used, referred, and transformed. But using only the shape and neglecting the parameterization of a curve does only make sense if the shape at least fulfills the properties of a one-dimensional manifold. Unfortunately, as can be seen in figure 2.1, the shape of parameterized curves may exhibit more general structures, such

as sharp edges and self-intersections, and, thus, does not necessarily abide to these constraints. As a consequence, more regular and specialized classes of curves need to be looked at. (Carmo 2016; Goldhorn, Heinz, and Kraus 2009; Kühnel 2013)

DEFINITION 2.2: Regular and Simple Parameterized Curves

Let γ be a parameterized curve. Then γ is said to be simple if it is injective and regular if the following statement holds.

$$\forall t \in \mathcal{D}(\gamma)^\circ \colon \quad \gamma'(t) \neq 0$$

Furthermore, if $\|\gamma'\| = 1$ then γ is parameterized by arc length.

The definition states that the derivative of a regular curve is apart from its boundary never zero and therefore a map of full-rank at every inner point. Hence, a regular curve is an immersion into Euclidean space and, as a result, its shape is an immersed submanifold that still may contain self-intersections. Using a curve that is parameterized by arc length is hereby only a way of defining a canonical parameterization for regular curves that will be used to simplify the definition of curvature. It is clear by definition, that every curve parameterized by arc length is also a regular curve. (Carmo 2016; Goldhorn, Heinz, and Kraus 2009; Kühnel 2013)

In applications that are able to handle self-intersections, regular curves are already sufficiently constrained. Indeed, a robust and stable algorithm for curve smoothing should be able to cope with self-intersections and even remove them if necessary. Still, to properly understand the transition, a regular curve shall become an embedding, such that, according to section A, its shape is actually a one-dimensional manifold. To state that a curve is also simple, ensures it exhibits no self-intersections other than the single intersection for closed curves. The results of the statements and thoughts of this and the previous paragraph are rigorously summarized in the following proposition. As this statement is a direct corollary, no additional rigorous proof will be given for it. (Carmo 2016; Goldhorn, Heinz, and Kraus 2009; Kühnel 2013)

COROLLARY 2.1: Simple, Regular Curves are Embeddings

Let γ be a smooth parameterized curve that is simple and regular. Then γ is a smooth embedding into Euclidean space and its image is an orientable and connected one-dimensional submanifold with boundary. If γ is closed, then its image has no boundary.

The fundamental part of the concept of geodesics is the curvature. To gain an intuitive approach to a definition of curvature of a regular curve, we need to be able to evaluate its length and arc length in Euclidean space. These will allow us to properly provide a canonical parameterization to overall simplify the use of regular curves.

DEFINITION 2.3: Length of Curves

Let γ be a parameterized curve on $[a, b]$ or $\mathbb{T}^1[a, b]$. Then the length $L(\gamma)$ and arc length, given by $s_\gamma \colon [a, b] \to [0, L(\gamma)]$ or $s_\gamma \colon \mathbb{T}^1[a, b] \to \mathbb{T}^1[0, L(\gamma)]$, respectively, of γ are defined by the following expressions.

$$L(\gamma) \coloneqq \int_a^b \|\gamma'(t)\| \, \mathrm{d}t \qquad s_\gamma(t) \coloneqq \int_a^t \|\gamma'(x)\| \, \mathrm{d}x$$

The definition of the length and arc length mainly stems from the physical interpretation that a parameterized curve is a trajectory in space over time of a particle moving with a specific velocity that is given by the derivative of the curve's parameterization. For evaluating the distance covered, one needs to integrate over the speed, that is given by the magnitude of the velocity, at each point in time.

For every parameterized curve γ, the length and arc length are well-defined because the function $\|\gamma'\|$ is continuous on a compact set and, consequently, uniformly continuous. Hence, the integral expressions defining $L(\gamma)$ and s_γ are finite and well-behaved. Furthermore, according to the fundamental theorem of calculus (Elstrodt 2018; Forster 2016), s_γ is continuously differentiable with derivative $s'_\gamma = \|\gamma'\|$ and surjective. A direct consequence for curves parameterized by arc length is that their length can be simply computed by $L(\gamma) = b - a$. This simplification leads to the idea to reparameterize general regular curves, such that they become curves parameterized by arc length and, accordingly, inheriting their good properties. The details for this approach are given by the following lemma for open parameterized curves as closed curves can be seen as a special kind of open curve.

LEMMA 2.2: Regular curves can be parameterized by arc length

Let $n \in \mathbb{N}$, $k \in \mathbb{N}_\infty$, $[a,b] \subset \mathbb{R}$ be a compact interval, and $\gamma\colon [a,b] \to \mathbb{R}^n$ be an n-dimensional parameterized curve of class C^k that is regular. Then up to a constant shift, there exists a unique k-times continuously differentiable and bijective function $u\colon [c,d] \to [a,b]$ on a compact interval $[c,d] \subset \mathbb{R}$ with strictly positive derivative, such that the composition $\gamma \circ u$ is a curve parameterized by arc length.

PROOF:
The restriction of $\|\cdot\|$ to $\mathbb{R}^n \setminus \{0\}$ is an infinitely differentiable function. The regularity of γ implies that $\|\gamma'\|$ and, as a direct consequence, the derivative of s_γ are strictly positive on (a,b). Hence, s_γ is bijective with k-times continuously differentiable inverse. The derivative of s_γ^{-1} must also be strictly positive in the interior and continuous as a composition of continuous functions.

$$\left(s_\gamma^{-1}\right)' = \frac{1}{s'_\gamma \circ s_\gamma^{-1}} = \frac{1}{\|\gamma'\| \circ s_\gamma^{-1}} > 0$$

Define $\tilde{\gamma} := \gamma \circ s_\gamma^{-1}$. Then by differential calculus, it follows, that $\tilde{\gamma}$ is k-times continuously differentiable and parameterized by arc length.

$$\left\|\tilde{\gamma}'\right\| = \left\|\gamma' \circ s_\gamma^{-1} \cdot \left(s_\gamma^{-1}\right)'\right\| = \left\|\frac{\gamma' \circ s_\gamma^{-1}}{s'_\gamma \circ s_\gamma^{-1}}\right\| = \left\|\frac{\gamma' \circ s_\gamma^{-1}}{\|\gamma'\| \circ s_\gamma^{-1}}\right\| = \frac{\left\|\gamma' \circ s_\gamma^{-1}\right\|}{\left\|\gamma' \circ s_\gamma^{-1}\right\|} = 1$$

This shows that s_γ^{-1} fulfills the required properties and the existence of a parameterization by arc length. To show its uniqueness up to a constant shift, assume an arbitrary function u as given in the lemma. Applying the same reasoning as for s_γ, it is clear that u^{-1} also is k-times continuously differentiable with strictly positive derivative on the interior. By calculation, we may then show its connection to the arc length.

$$1 = \left\|(\gamma \circ u)'\right\| = \left\|\gamma' \circ u\right\| \cdot u' = \frac{\|\gamma' \circ u\|}{(u^{-1})' \circ u} = \frac{\|\gamma'\|}{(u^{-1})'} \quad \Longrightarrow \quad \|\gamma'\| = \left(u^{-1}\right)'$$

To integrate this formula, let $t \in [a,b]$ be arbitrary and use again the fundamental theorem of calculus.

$$s_\gamma(t) = \int_a^t \left\|\gamma'(x)\right\| \,\mathrm{d}x = \int_a^t \left(u^{-1}\right)'(x) \,\mathrm{d}x = u^{-1}(t) - u^{-1}(a) = u^{-1}(t) - c$$

$$u = s_\gamma^{-1} \circ s_\gamma \circ u = s_\gamma^{-1}(\cdot - c) \circ u^{-1} \circ u = s_\gamma^{-1}(\cdot - c)$$

This shows that, up to a constant shift, u is the same as s_γ^{-1} and the uniqueness of the parameterization by arc length. □

The proof above shows, that the inverse of the arc length of a regular curve up to some shifting can be used as a unique reparameterization to transform it into a curve parameterized by arc length. In this sense, the lemma makes clear that it is sufficient to handle curves that are parameterized by arc length instead of general regular curves. According to this, we will define this as the canonical parameterization of a regular curve.

DEFINITION 2.4: Canonical Parameterization by Arc Length

Let γ be a regular curve. Then its canonical parameterization by arc length $\bar{\gamma}$ is defined by the following expression.

$$\bar{\gamma} \coloneqq \gamma \circ s_\gamma^{-1}$$

For smooth curves parameterized by arc length, the definition of their curvature is now straightforward, as we understand it as the amount the curve bends at a point when moving along its trajectory. Referring to differential calculus, the magnitude of the second derivative exactly describes this specific amount. (Forster 2016; Goldhorn, Heinz, and Kraus 2009)

DEFINITION 2.5: Curvature of Regular Curves

Let $k \in \mathbb{N}_\infty$ with $k \geq 2$, γ be a curve parameterized by arc length of class C^k, and φ be a regular curve of class C^k. Their respective curvatures $\kappa(\gamma)$ and $\kappa(\varphi)$ are defined by the following expressions.

$$\kappa(\gamma) \coloneqq \|\gamma''\| \qquad \kappa(\varphi) \coloneqq \kappa(\bar{\varphi}) \circ s_\varphi$$

The definition involves second-order derivatives and therefore at least requires the curves to be of class C^2. In differential geometry, only smooth curves are typically observed where this states no further constraints (Goldhorn, Heinz, and Kraus 2009). For general regular curves, the tangent vector is not normalized which results in a scaling of the curvature value. The definition omits this peculiarity by using the arc length function to get back to the original parameterization.

To finally provide a definition for geodesics, it is no longer sufficient to look at smooth curves freely traversing the Euclidean space. Instead, we will embed curves into smooth surfaces which already exhibit intrinsic curvature. Based on this intrinsic curvature, the overall curvature of a curve can be decomposed into the geodesic and normal curvature. Hereby, the geodesic curvature describes the amount of curvature an intrinsic observer of the ambient surface would measure walking along the curve without knowledge of the extrinsic curvature. The normal curvature on the other hand is only observable from the ambient space and depends on the extrinsic curvature and the tangential vector of the curve. In the context of curve smoothing, the surfaces, we are looking at, are given by compact (orientable) Riemannian submanifolds (with boundary) of dimension two embedded in three-dimensional Euclidean space. (Carmo 2016; Goldhorn, Heinz, and Kraus 2009; Kühnel 2013)

DEFINITION 2.6: Geodesic and Normal Curvature of Curves

Let M be a Riemannian submanifold embedded in Euclidean space and γ be a curve embedded in M and parameterized by arc length. The geodesic curvature

$\kappa_{\mathrm{g}}(M,\gamma)$ and the normal curvature $\kappa_{\mathrm{n}}(M,\gamma)$ are defined as follows.

$$\bar{\kappa}_{\mathrm{g}}(M,\gamma) \coloneqq \left\| P_{\mathrm{T}_\gamma M}(\gamma'') \right\| \qquad \bar{\kappa}_{\mathrm{n}}(M,\gamma) \coloneqq \left\| P_{\mathrm{N}_\gamma M}(\gamma'') \right\|$$

Geodesic and normal curvature are simply defined by computing the norm of projecting the direction of bending into the tangential and normal space of the manifold, respectively. Hence, this definition is consistent with the interpretation given above. As a direct consequence of the definition, the following statement can be deduced without further proof.

COROLLARY 2.3: Relationship of Geodesic and Normal Curvature

Let M be a Riemannian submanifold embedded in Euclidean space and γ be a curve embedded in M and parameterized by arc length. Then the following holds.

$$\kappa^2(\gamma) = \bar{\kappa}_{\mathrm{g}}^2(M,\gamma) + \bar{\kappa}_{\mathrm{n}}^2(M,\gamma)$$

Up to now, all curvature definitions have been non-negative scalar functions that did not provide any orientation. But in the context of orientable Riemannian submanifolds of dimension two, further knowledge about the direction of geodesic bending can be provided (Goldhorn, Heinz, and Kraus 2009). This is especially useful for implementation-specific work. Providing orientations for manifolds and curves allows for a more efficient moving of points along a curve or surface and access to its neighbors.

DEFINITION 2.7: Oriented Curvatures of Curves

Let M be a two-dimensional oriented Riemannian submanifold embedded in $\mathbb{R}^3$ and γ be a curve embedded in M and parameterized by arc length.
Furthermore, let $\nu\colon M \to \mathrm{N}M$ be a differentiable field of positively oriented unit normals and $\mu\colon M \to \mathrm{T}M$ be a differentiable field of normalized tangent vectors such that all values of $(\gamma', \mu \circ \gamma)$ are a positively oriented basis.
The oriented geodesic and normal curvature are defined as follows, respectively.

$$\kappa_{\mathrm{g}}(M,\gamma) = \langle \mu \circ \gamma \,|\, \gamma'' \rangle \qquad \kappa_{\mathrm{n}}(M,\gamma) = \langle \nu \circ \gamma \,|\, \gamma'' \rangle$$

The oriented geodesic curvature is positive when the curve bends to the left and negative when it bends to the right with respect to the chosen positive orientation of the ambient submanifold. The oriented normal curvature is not as important in our context as it only describes the bending of the surface along the trajectory. Despite this, the previous corollary still holds for these oriented formulations.

For an intrinsic observer positioned in the ambient submanifold, the curve will be straight if it does not exhibit any geodesic curvature, because then it neither bends to the right nor to the left at any point. This is one of the characterizing properties of geodesics in classical and modern differential geometry and, for us, will serve as a definition and generalization (Carmo 2016; Goldhorn, Heinz, and Kraus 2009; Kühnel 2013; Polthier and Schmies 2006).

DEFINITION 2.8: Geodesic

Let M be a Riemannian submanifold embedded in Euclidean space and γ be a curve embedded in M and parameterized by arc length. Then γ is called a geodesic if $\bar{\kappa}_{\mathrm{g}}(M,\gamma) = 0$.

The problem of finding a geodesic for a given ambient submanifold, a starting point, and a starting direction is called the geodesic initial value problem. In a rigorous formulation, it breaks down to a system of second-order ordinary differential equations for which it can be proven that a locally unique solution exists in some sense (Polthier and Schmies 2006). If no starting direction is given, but instead the point to end at, then we call it the geodesic boundary value problem. The boundary value problem is not as easy to handle as the initial value problem. There is typically no unique solution despite the fact that there may be no solutions at all (Polthier and Schmies 2006). Alas, for smoothing already given curves, the boundary value problem needs to be solved in a way that we find a single solution if there exists at least one. (Carmo 2016; Goldhorn, Heinz, and Kraus 2009; Kühnel 2013)

Another important alternative characterization states that geodesics are a critical point of their length functional with respect to variations tangential to the ambient submanifold (Polthier and Schmies 2006). In general, this is understood to mean that a curve is locally shortest, that is, a slight change of the trajectory at any point will only increase the overall length of the curve. Unfortunately, a critical point is not necessarily a local minimum of the length of a curve and by no means a global minimum. Working with smooth submanifolds, this property is equivalent to our defining geodesic property. Embedding a curve in a polyhedral non-differentiable submanifold allows for multiple non-trivial generalizations of geodesics, namely straightest and shortest discrete geodesics, based on each characterizing property as these are no longer equivalent in such a context (Polthier and Schmies 2006).

2.2 Polyhedral Surfaces

There are definitions for manifolds with corners. In the topological case, these are equivalent to manifolds with boundaries. For the smooth case, it is different. The half space is only homeomorphic and not diffeomorphic to the cube space. so, we could generalize a polyhedral surface as a smooth manifold with corners. on the definitions, there is no uniform agree. So, we will not use the generalization of manifolds with corners. we will use the theory of polthier which should be applicable to manifolds with corners.

DEFINITION 2.9: Triangle

Let $n \in \mathbb{N}$ with $n \geq 2$ and $A, B, C \in \mathbb{R}^n$ affinely independent points. Then the (topological) triangle $\triangle$ (embedded in $\mathbb{R}^n$) is given by the following expression.

$$\triangle := \{uA + vB + wC \mid u, v, w \in [0,1], u + v + w = 1\}$$

The triangle's vertices $\mathcal{V}(\triangle)$ and edges $\mathcal{E}(\triangle)$ are hereby defined as follows.

$$\mathcal{V}(\triangle) := \{A, B, C\} \qquad \mathcal{E}(\triangle) := \left\{\overline{AB}, \overline{BC}, \overline{CA}\right\}$$

$$\partial\triangle := \bigcup \mathcal{E}(\triangle)$$

$$\Pi := \{\pi \colon \{1,2,3\} \to \mathcal{V}(\triangle) \mid \pi \text{ bijective}\}$$

$$D := \left\{(u,v) \in [0,1]^2 \;\middle|\; u + v < 1\right\}$$

$$\varphi_\pi \colon \overline{D} \to \triangle \qquad \varphi_\pi(u,v) := (1-u-v)\pi(1) + u\pi(2) + v\pi(3)$$

$$\pi \sim \pi' :\iff \exists \sigma \in \mathrm{S}_3, \operatorname{sgn}\sigma = 1 \colon \pi' = \pi \circ \sigma$$

$$\Pi/\sim = \{[\pi], [\bar{\pi}]\}$$

$$\triangle_{[\pi]} \coloneqq (\triangle, [\pi])$$

$$\mathcal{E}(\triangle_{[\pi]}) \coloneqq \{\overrightarrow{\pi_1\pi_2}, \overrightarrow{\pi_2\pi_3}, \overrightarrow{\pi_3\pi_1}\}$$

$$\vartheta_{\triangle}(A) = \arccos \frac{\left\langle \overrightarrow{AB} \,\middle|\, \overrightarrow{AC} \right\rangle}{\left\| \overrightarrow{AB} \right\| \left\| \overrightarrow{AC} \right\|}$$

A triangle is an orientable topological 2-manifold with boundary embedded in $\mathbb{R}^n$. The atlas is given by $\left\{ \left(\varphi_\pi(D), \varphi_\pi|_D^{-1}\right) \mid \pi \in \Pi \right\}$ The inner set $\triangle^\circ$ is an orientable smooth 2-manifold without boundary. Its atlas is given by the restrictions $(\triangle^\circ, \varphi_\pi|_{D^\circ}^{-1})$. Triangles are planar surfaces and, hence, geodesics are given by straight lines. Furthermore, the triangle is by definition a convex set and therefore always provides a unique solution for the geodesic boundary value problem.

DEFINITION 2.10: Polyhedral Surface and Surface Mesh

Let $n \in \mathbb{N}$ with $n \geq 2$ and $\mathcal{T} \neq \emptyset$ be a finite set of triangles embedded in $\mathbb{R}^n$. Let $S = \bigcup \mathcal{T}$ be a two-dimensional topological manifold (with boundary), such that for all $\triangle_1, \triangle_2 \in \mathcal{T}$ with $\triangle_1 \neq \triangle_2$ the following holds.

$$\triangle_1 \cap \triangle_2 \in \{\emptyset\} \cup [\mathcal{V}(\triangle_1) \cap \mathcal{V}(\triangle_2)] \cup [\mathcal{E}(\triangle_1) \cap \mathcal{E}(\triangle_2)]$$

In this case, S is called a (topological) polyhedral surface (embedded in $\mathbb{R}^n$). With $\mathcal{V}(S)$ we denote its vertices, with $\mathcal{E}(S)$ its edges, and with $\mathcal{F}(S)$ its faces.

$$\mathcal{V}(S) \coloneqq \bigcup_{\triangle \in \mathcal{T}} \mathcal{V}(\triangle) \qquad \mathcal{E}(S) \coloneqq \bigcup_{\triangle \in \mathcal{T}} \mathcal{E}(\triangle) \qquad \mathcal{F}(S) \coloneqq \mathcal{T}$$

(topological) surface mesh.

$$\mathcal{M}(S) \coloneqq \bigcup \mathcal{E}(S)$$

$$\partial\mathcal{E}(S) \coloneqq \{e \in \mathcal{E}(S) \mid \exists! \triangle \in \mathcal{F}(S) \colon e \subset \triangle\} \qquad \mathcal{E}^\circ(S) \coloneqq \mathcal{E}(S) \setminus \partial\mathcal{E}(S)$$

$$\partial S \coloneqq \bigcup \partial\mathcal{E}(S) \qquad \partial\mathcal{V}(S) \coloneqq \mathcal{V}(S) \cap \partial S \qquad \mathcal{V}^\circ(S) \coloneqq \mathcal{V}(S) \setminus \partial\mathcal{V}(S)$$

A surface mesh is again a topological 2-manifold.

DEFINITION 2.11: Oriented Surface Mesh

Let S be a surface mesh and $\Pi \colon \mathcal{F}(S) \to \bigsqcup_{\triangle \in \mathcal{F}(S)} \Pi_\triangle$.

$$\forall \triangle_1, \triangle_2 \in \mathcal{F}(S), \triangle_1 \neq \triangle_2 \colon \quad \mathcal{E}(\Pi(\triangle_1)) \cap \mathcal{E}(\Pi(\triangle_2)) = \emptyset$$

(S, Π) is called an oriented surface mesh.

DEFINITION 2.12: Discrete Surface Mesh Curve

Let $n \in \mathbb{N}$, S be a surface mesh, and $x \in \left[\bigcup \mathcal{E}(S)\right]^{n+1}$ be a tuple of $n+1$ points

on the edges of the surface mesh S, such that adjacent points lie on the same triangle.

$$\forall k \in \mathbb{N}, k \leq n \colon x_k \neq x_{k+1} \wedge \exists \triangle \in \mathcal{F}(S) \colon x_k, x_{k+1} \in \triangle$$

The (topological) curve given by connecting adjacent points by a straight line is called the (topological) discrete surface mesh curve.

DEFINITION 2.13: Surface Mesh Curve without Artifacts

Let S be a polyhedral surface and γ be a surface mesh curve characterized by $x \in \mathcal{M}^{n+1}(S)$.

$$\forall k \in \mathbb{N}, 1 \leq k \leq n \colon \mathcal{F}_S(x_{k-1}) \cap \mathcal{F}_S(x_{k+1}) = \emptyset$$

If γ is a closed curve, that is $x_0 = x_{n+1}$, then

$$\mathcal{F}_S(x_1) \cap \mathcal{F}_S(x_n) = \emptyset$$

Surface mesh curves without artifacts will never contain points that lie on the inside of a boundary edge. They may contain boundary vertices.

DEFINITION 2.14: Total Vertex Angle

$$\vartheta_S(v) \coloneqq \sum_{\triangle \in \mathcal{T}, v \in \triangle} \vartheta_\triangle(v)$$

DEFINITION 2.15: Curve Angle

Let S be a polyhedral surface and $p, x, q \in \mathcal{M}(S)$, $x \in S^\circ$ be adjacent vertices of surface mesh curve without artifacts on S. Let $\mathcal{V}_S(x)$ the neighboring vertices. The neighbor set can be partitioned into three possible empty tuples, such that $p, (x), q$ is surface mesh curvature without artifacts. There exists two tuples $(p = x_{\lambda(0)}, x_{\lambda(1)}, \ldots, x_{\lambda(n)}, x_{\lambda(n+1)} = q)$ and $(p, x_{\rho(1)}, \ldots, x_{\rho(m)}, q)$ that characterize a surface mesh curve without artifacts.

$$\beta_\lambda = \sum_{k=1}^{n} \sphericalangle(x_{\lambda(k-1)}, x_{\lambda(k)}, x_{\lambda(k+1)}) \qquad \beta_\rho = \sum_{k=1}^{m} \sphericalangle(x_{\rho(k-1)}, x_{\rho(k)}, x_{\rho(k+1)})$$

The (unoriented) curve angle at x is given by the sum

$$\beta = \min\{\beta_\lambda, \beta_\rho\}$$

If there is an orientation Π for S then the (oriented) curve angle is given by the segmentation which points are connected by directed edge paths in their respective triangles such that x is not contained.

For the oriented case, the curve angle can only be defined for curve vertices that do not lie

on the boundary of S. Also the interpretation of the total vertex angle for boundary points is different. This makes a generalization of the oriented discrete curvature at boundary points difficult.

DEFINITION 2.16: Discrete Geodesic Curvature

$$\kappa_{\mathrm{g}}(\gamma) = \pi - \frac{2\pi}{\vartheta_S(\gamma)}\beta_S(\gamma)$$

DEFINITION 2.17: Discrete Geodesic Curvature for Boundaries

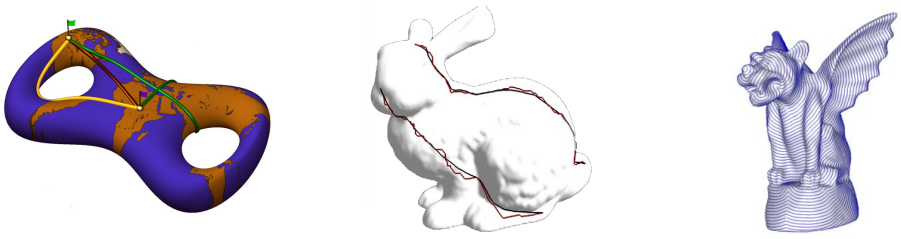
DEFINITION 2.18: Discrete Geodesic Problems

LEMMA 2.4: Straightest and shortest geodesics are surface mesh curves without artifacts.

3 Related Work

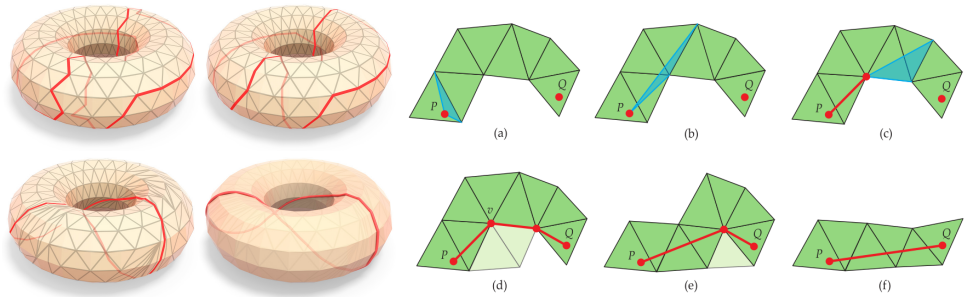
As marked in the introduction in section 1, the smoothing of curves on surface meshes is an essential operation for mesh processing and, as a consequence, for many other domain areas, like computer graphics, image-based medicine, and engineering, that rely on such tools (Ji et al. 2006; Kaplansky and Tal 2009). In most of its applications, initial curves are either provided by means of direct user interaction or by automatic or semiautomatic feature detection algorithms (Lawonn et al. 2014; Zachow et al. 2003). The finite precision of the underlying surface mesh together with all the steps included to define an initial curve usually makes resulting lines contain non-smooth artifacts which may violate given constraints or expected properties and therefore degrade its quality (Kaplansky and Tal 2009; Lawonn et al. 2014). Introducing a smoothing stage into the curve processing pipeline, the mesh segmentation is expected to be of much higher quality which greatly increases its usage for areas like machine learning (Benhabiles et al. 2011) or medicine (Alirr and Abd. Rahni 2019; Zachow et al. 2003). During the last two decades, there have been multiple successful attempts for constructing algorithms to smooth curves on surfaces (Bischoff, Weyand, and Kobbelt 2005; Hofer and Pottmann 2004; Lawonn et al. 2014; Mancinelli et al. 2022). In this section, a brief overview of their major contributions is given.

As stated in the previous section 2, a crucial tool for working with curves on two-dimensional manifolds is the ability to find and generate geodesic paths on triangle meshes in the sense of the initial and boundary value problem. The rigorous mathematical concepts and definitions for the discrete geodesics problems have been elaborated by Mitchell, Mount, and Papadimitriou (1987) and Polthier and Schmies (2006) first published in 1997. It was marked that the generalization of the concept of geodesics from smooth manifolds to polyhedral surfaces is not unique. By introducing the notion of discrete geodesic curvature, Polthier and Schmies (2006) instead fleshed out two main definitions for discrete geodesics on polyhedral surfaces, namely locally shortest geodesics and straightest geodesics. Both of these concepts coincide on smooth manifolds but differ on polyhedral surfaces resulting in varying solutions to the discrete geodesics problems. Additionally, Mitchell, Mount, and Papadimitriou (1987) built an algorithm to solve the discrete boundary value problem for locally shortest geodesics, that uses a continuous version of the algorithm of Dijkstra (1959) to find the shortest path connecting two given points. Furthermore, Polthier and Schmies (2006) provided an iterative algorithm to solve the discrete initial value problem of finding the unique straightest geodesic



(a) *Polthier and Schmies (2006)* **(b)** *Martínez, Velho, and Carvalho (2005)* **(c)** *Surazhsky et al. (2005)*

Figure 3.2: ***Examples of Discrete Geodesics***
The images show examples of the different approaches for the generation of geodesics on polyhedral surfaces described by Polthier and Schmies (2006), Martínez, Velho, and Carvalho (2005), and Surazhsky et al. (2005). All the images have been provided by the authors themselves.



(a) *Sharp and Crane (2020)* **(b)** *Mancinelli et al. (2022)*

Figure 3.3: ***State-of-the-Art Discrete Geodesic Tracing***

The left images shows the approaches for the computation and tracing of geodesics on polyhedral surfaces described by Sharp and Crane (2020). The right image shows the funnel algorithm scheme of Mancinelli et al. (2022). All the images have been provided by the authors themselves.

given a starting point and a direction for which an example can be seen in figure 3.2a. Hereby, they have proven its correctness and also introduced the notion for parallel translation of vectors along polyhedral surfaces for particle transportation and differential equations.

Based on the results of Mitchell, Mount, and Papadimitriou (1987), Kimmel and Sethian introduced the so-called fast marching approach (FMM) to efficiently approximate the shortest geodesic connecting two given points, which used the eikonal equation to build propagating fronts and therefore more efficiently generate distance fields (Kimmel and Sethian 1996, 1998; Sethian 1996). Surazhsky et al. (2005) followed with a formulation of exact and approximate algorithms to solve the discrete initial and boundary value problem, which could be evaluated efficiently by the use of such distance fields. Figure 3.2c shows an example of their work. Extending the idea of distance fields as an intermediate step to the generation of geodesics, Crane, Weischedel, and Wardetzky (2013) also used the gradient of the heat kernel to reconstruct a distance field by solving the Poisson equation and Bommes and Kobbelt (2007) generalized the algorithm of Surazhsky et al. (2005) to not only handle isolated starting points but also general starting polygons on polyhedral surfaces.

Building on the theory of Polthier and Schmies (2006), Martínez, Velho, and Carvalho (2005) provided an algorithm to solve the discrete boundary value problem for locally shortest geodesics. Using paths generated by FMM as initial curve for their algorithm, they were able to iteratively improve this curve with an optimization approach making it eventually converge to the exact solution. An example of their result has been visualized in figure 3.2b. Improving the idea of Martínez, Velho, and Carvalho (2005), Xin and Wang (2007) compared various methods to provide an initial path and presented a visibility-based algorithm to determine an exact locally shortest path on polyhedral surfaces after finitely many steps. Emphasizing this discrete nature of the problem, Sharp and Crane (2020) showed that it is possible to find exact geodesic paths for triangle meshes just by intrinsically flipping the edges of the surface mesh and constructed an algorithm, named *FlipOut*, which finds the locally shortest geodesic in the same isotopy class as the initial curve and that is guaranteed to terminate after a finite number of steps. The algorithm never needs to embed the final, flipped triangulation into the surface again and exhibits real-time performance even for millions of triangles. One can see an example of the *FlipOut* scheme in figure 3.3. To further improve the computation of locally shortest geodesics in a finite number of steps, Mancinelli et al. (2022) combined the

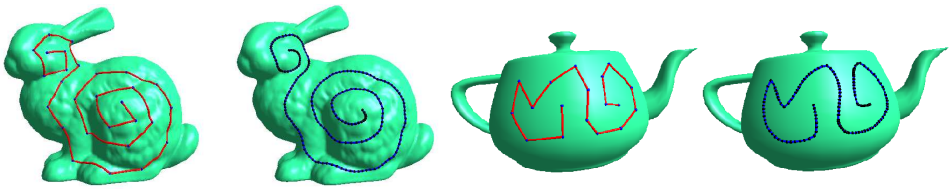


Figure 3.4: ***Examples of Subdivision Curves by Morera, Velho, and Carvalho (2008)***
The images show examples of the curve smoothing approach given by Morera, Velho, and Carvalho (2008) which used subdivision schemes. All images have been provided by the authors themselves. Red lines indicate the initially chosen curve and blue lines the resulting smoothed curve.

funnel algorithm of Lee and Preparata (1984) with the ideas of Xin and Wang (2007) into a three-stage algorithm that surpasses *FlipOut* due to its superior initial extraction of a triangle strip that results from using an optimized shortest path algorithm on the dual graph of the surface mesh. For the initial strip, the shortest path algorithm uses an A*-like distance metric and an underlying double ended queue driven by the small-label-first and large-label-last heuristics to avoid the use of a priority queue. This scheme is also illustrated in figure 3.3. Additional literature covering topics concerned with discrete geodesics on polyhedral surfaces is also well-covered by Crane et al. (2020).

For the actual creation of smooth curves on surfaces, evidence shows that only a few main approaches have emerged. Presumably, the most intuitive way for a curve smoothing algorithm to work is by using subdivision schemes for polygonal lines, also called corner cutting. The algorithm thereby subdivides each line segment and positions newly created points in such a way that the resulting curve is smoother than the previous one. First introduced and used in the planar case by Chaikin (1974) and Dyn, Levin, and Liu (1992), Morera, Velho, and Carvalho (2008) generalized the algorithm to polygonal lines on surfaces. Two examples of this can be seen in figure 3.4. Unfortunately, the subdivision curve may consist of points located anywhere inside the faces of the underlying mesh. Hence, its trajectory is not a surface curve in the strong rigorous mathematical sense and might miss essential parts of the mesh. The objective to construct a robust curve smoothing algorithm dictates that for discrete surfaces all line segments should lie inside a face of the surface.

The smoothing of curves based on features of the surface mesh for automatic mesh segmentation and cutting has been shown to successfully work by Jung and Kim (2004),

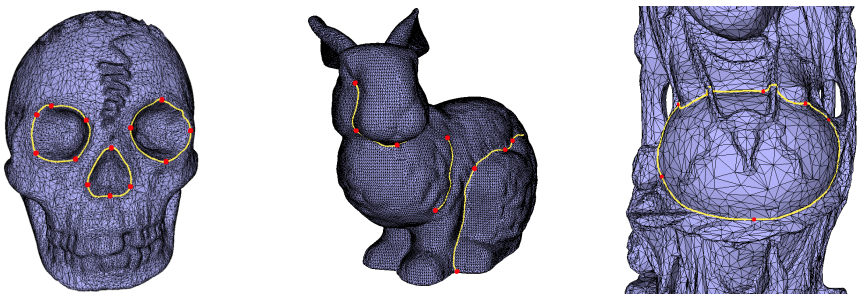


Figure 3.5: ***Examples of Snakes by Lee and Lee (2002)***
The images show examples of the curve smoothing approach given by Lee and Lee (2002) which used a generalization of snakes to detect surface features. All images have been provided by the authors themselves.

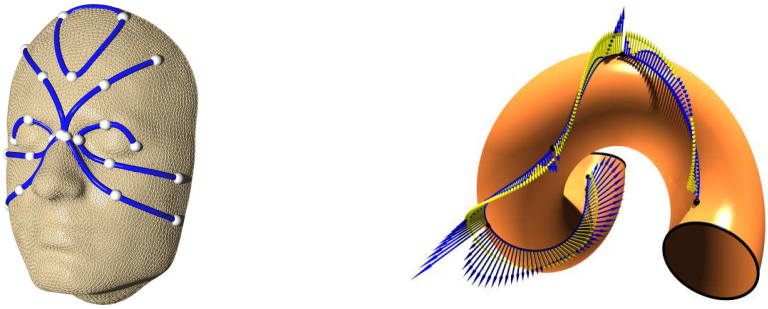


Figure 3.6: ***Examples of Energy-Minimizing Splines by Hofer and Pottmann (2004)***
The images show examples of the design of smooth curves given by Hofer and Pottmann (2004) that used a variational approach to minimize an energy-like functional on the curve. All images have been provided by the authors themselves.

Bischoff, Weyand, and Kobbelt (2005), and Lai et al. (2007). To classify surface features, Lai et al. (2007) used a feature-sensitive curve smoothing which allowed them to obtain smooth boundaries for mesh features. Both publications, Jung and Kim (2004) and Bischoff, Weyand, and Kobbelt (2005), are building upon the previous work of Lee and Lee (2002) and Lee et al. (2004). They generalized so-called snakes for two-dimensional manifolds to represent curves on surfaces that are able to find crucial mesh features by providing an initial curve. Figure 3.5 provides some examples of the results achieved by Lee and Lee (2002). First introduced by Kass, Witkin, and Terzopoulos (1988) for two-dimensional images, snakes are closed curves that evolve over many iterations to the features of the mesh by minimizing internal and external forces based on curvature, length, and distance to features. For snakes, the initial shape is completely unimportant. They are allowed to merge or split, such that a rapid movement towards the features of the mesh is to be expected. As a direct consequence, feature-based curve smoothing does not allow for arbitrary trajectories and would lead to a curve that may not be assumed to be near the original curve, which makes these algorithms bad candidates for general curve smoothing.

Another approach for representing and generating smooth curves on surfaces is through the use of splines. Hofer and Pottmann (2004) and Pottmann and Hofer (2005) determined splines in general manifolds by addressing the design of curves as an optimization problem in the sense of minimizing the curve's overall quadratic energy and using a variational approach to compute a solution. Their approach is not only applicable to curves on surface meshes but can be used for a much broader variety of cases, including for example the design of rigid body motions. Some of their results are shown in figure 3.6. Alas, for a small number of control points, the resulting curve may still not be assumed to exhibit a close distance to the initially selected curve. Overcoming this issue would involve adding many more control points and, eventually, a much higher burden for the user who would need to define those points. According to Mancinelli et al. (2022), the variational approach to solve the optimization problem for the design of curves is also expected to provide a poor performance for surface meshes that consist of millions of triangles.

These results quickly lead to the representation of smooth curves by using generalized Bézier splines. Two essential contributions are given by Martínez, Carvalho, and Velho (2007) and Mancinelli et al. (2022). They lift the concept of Bézier curves in the two-dimensional Euclidean space to geodesic Bézier splines located in the surface. The initially chosen

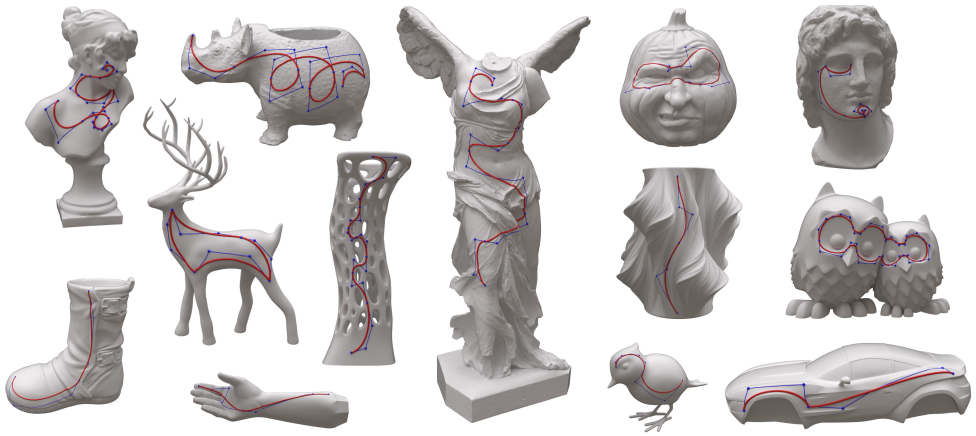


Figure 3.7: ***Examples of Bézier Splines by Mancinelli et al. (2022)***
The images show examples of the design of smooth curves given by Mancinelli et al. (2022) that used generalized Bézier splines to represent a surface curve. Blue lines indicate the initially chosen curve and red lines the resulting spline-based curve. All images have been provided by the authors themselves.

curve samples are thereby used as control points to determine the individual shapes of the Bézier splines. Mancinelli et al. (2022) successfully showed their approach to be superior to other spline alternatives and provided real-time performance when tracing the trajectories of the given splines on the surface for even high-resolution meshes. In addition, they offer a robust implementation of their algorithm in an open-source C++ framework, named *Yocto/GL* (Pellacini, Nazzaro, and Carra 2019), that can be found on GitHub[3]. The framework is a rather large codebase and a specific documentation for the code responsible for generating, tracing, and controlling Bézier splines could not be found, though. A whole gallery of examples that has been provided by Mancinelli et al. (2022) can be seen in figure 3.7. Nevertheless, their approach exhibits similar issues compared to the variational spline approach in that only the use of many control points will make sure that the resulting curve will be near to the initially defined curve. In this sense, the approach of Mancinelli et al. (2022) does not fit our need by being specialized for vector graphics on surface meshes.

Generalizing on the iterative algorithm of Martínez, Velho, and Carvalho (2005), Lawonn et al. (2014) created an algorithm for curve smoothing based on curvature values given for each trajectory point. Each iteration, the algorithm tries to locally fulfill the given curvature constraint, eventually converging to its final smooth curve. The algorithm guarantees a close distance to the initial curve and can also be used with a simplified user interaction where only one parameter has to be adjusted. During the process, all iterations adapt to the resolution of the underlying surface mesh and directly provide the curve on the surface without the need of tracing or projection. The algorithm was shown to be robust against geometric and parametric noise and applied in a medical context, where domain experts evaluated its usability. Lawonn et al. (2014) proved the convergence of the algorithm and also compared the quality of the generated smoothed curves against the spline-based variational approach without any issue. Figure 3.8 shows two examples for this algorithm that visualizes medicine-specific models in the context of mesh cutting. Furthermore, no surface normals or curvature, that would

[3] *Yocto/GL* (2023). *Tiny C++ Libraries for Data-Oriented Physically-based Graphics*. URL: https://github.com/xelatihy/yocto-gl (visited on 11/19/2022).

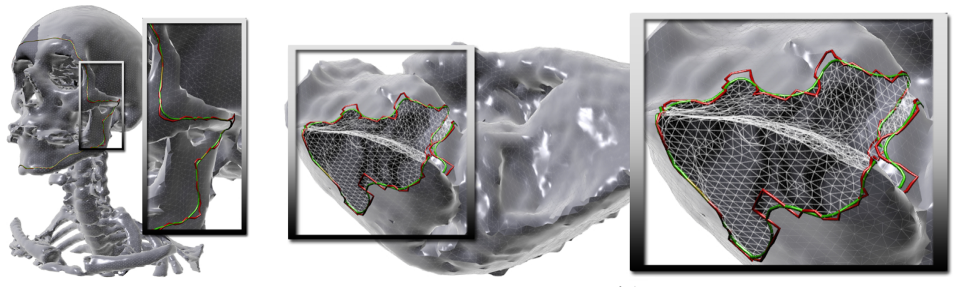


Figure 3.8: ***Examples of Curve Smoothing by Lawonn et al. (2014)***
The images show examples of the curve smoothing algorithm given by Lawonn et al. (2014) that uses a generalization of the algorithm given by Martínez, Velho, and Carvalho (2005) to fit curvature values. Red lines indicate the initially chosen curve and green lines the resulting smoothed curve. All images have been provided by the authors themselves.

need to be evaluated first, is needed for the algorithm. As a result, it may also be formulated for generalized two-dimensional triangular manifolds leaving a wide variety of mesh data structures to choose from (Guibas and Stolfi 1985). Unfortunately, Lawonn et al. (2014) do not provide any language-specific implementation or performance evaluation.

According to the explanations and descriptions above, for the purpose of this thesis, our design and implementation will mainly focus on the approach given by Lawonn et al. (2014). Their algorithm seems to fit our needs in nearly all important aspects. To solve the potential performance issue and get real-time behavior, a parallelization on the CPU and GPU will be carried out. We will also strive for an optimized curve initialization and geodesics generation that builds upon the basic building blocks of the approach given by Mancinelli et al. (2022).

4 Design

Experimenting with various designs and implementations of algorithms, that robustly and efficiently smooth discrete curves on surface meshes, requires us to use a whole program pipeline or framework for loading input data, intuitively working with user interaction, and visualizing any intermediate results. As changing specific parts of such a pipeline may affect the performance, outcomes, and overall behavior of an algorithm or program, a brief overview over all of its components is essential to allow for reproducible results and will therefore be given in the following subsections. This will also make it possible for readers to simply reconstruct, adjust, and improve the pipeline for their own domain-specific projects. Afterwards, I will thoroughly elaborate on the design and implementation of chosen data structures and algorithms together with their mathematical primitives for curve smoothing.

The pipeline or framework, described in this thesis, has been manually implemented with the C++ programming language and the OpenGL graphics API using some external libraries to handle low-level tasks. Already described in the introduction in section 1, this choice is well suited for open-source graphics applications and provides programmers with a large freedom when it comes to the implementation of data structures and algorithms. The complete source code of the project, called *nanoreflex*, is provided as an open-source repository on GitHub.

4.1 Overview of the Program Pipeline

In the following, the stages of the implemented program pipeline are stated and summarized. These stages naturally arise from an insight into the generation and tracing of geodesics on surface meshes, described in section 3. Locally shortest or straightest geodesics that follow as a solution from the discrete boundary value problem are in some sense the smoothest curves that cannot be made any smoother. As we typically need to provide an initial curve for those geodesics to be found, the whole generation and tracing of geodesics can be looked at as an extreme smoothing process. Adding parameters to let the user decide on when to stop this process then results in the pipeline given below and schematically shown in figure 4.9.

INSERT YOUR IMAGE HERE!

Figure 4.9: ***Program Pipeline Stages***
The scheme shows all the stages of the implemented program pipeline. The arrows indicate data flow and dependencies.

1. **Surface Mesh Loading:**
   Load a surface mesh given by a specific file format from the storage.
2. **Surface Mesh Preprocessing:**
   (a) Generate a connected mesh object to properly model the topology of the input.
   (b) Generate the pseudo-normals for each vertex.
   (c) Generate the dual graph for the neighbors of each triangle.
   (d) Check that the mesh is a valid orientable two-dimensional topological manifold.
3. **Initial Curve Selection:**
   Let the user choose an initial curve by drawing on the surface.

4. **Parameter Selection:**
   Let the user choose parameters for the curve smoothing algorithm.
5. **Curve Smoothing:**
   Smooth the initial curve according to the constraints given by the selected parameters.
6. **Postprocessing:**
   Optionally apply the smoothed curve in a domain-specific context.

According to figure 4.9, the output of every stage is not only forwarded to the next stage but also specifically visualized and rendered to the screen to provide the user with visual feedback and to catch errors or exceptional behavior as early as possible. Besides, additional user interactions will be processed for all stages to allow for measurements and adjustments and to be able to react to the output of previous stages. The loading of surface meshes from different file formats is typically handled by loader libraries, such as *Assimp*, and will not be further explained. On the other hand, surface mesh preprocessing and the initial curve selection are crucial steps in the whole pipeline which remove geometric degeneracies and artifacts that would otherwise violate assertions needed for the correct execution of the algorithm. Therefore both stages will be discussed in more detail in the following subsections. The postprocessing in our case is optional and only provided for the sake of completeness. The parameter selection is highly dependent on the implementation of the curve smoothing algorithm and, as a result, it will be described together with the curve smoothing itself.

4.2 Polyhedral Surface Data Structure

For loading and preprocessing a surface mesh, the data structure of a polyhedral surface has to be defined first. The easiest representation, especially useful for rendering, is to have two separate lists of vertices and faces. Each vertex should at least store its position but will most likely also provide a pseudo-normal for better visualization. Every face is a triangle, that refers to the three vertices it contains. This is the minimal information that needs to be provided to create a polyhedral surface.

To allow points to move on the surface, each face must know about its neighbors. Every triangle can at most have three neighbors. So, each triangle will also refer to its neighbors. When there is no neighbor, we will store an invalid id.

This makes the dual graph of the surface mesh available.

Code 4.1: Polyhedral Surface

```
struct polyhedral_surface {
  static constexpr uint32_t invalid = -1;

  struct vertex {
    vec3 position{};
    vec3 normal{};
  };
  using vertex_id = uint32_t;

  struct edge : std::array<vertex_id, 2> {};

  struct face : std::array<vertex_id, 3> {};
  using face_id = uint32_t;
```

```
  struct face_neighbors : std::array<face_id, 3> {};

  std::vector<vertex> vertices{};
  std::vector<face> faces{};
  std::vector<face_neighbors> dual_graph{};
};
```

An optional step would also to construct the vertex graph. But we probably do not need it.

The triangle-based data structured described so far was also used by Mancinelli et al. (2022) and seems to be superior to other edge-based data structures. Looking at Delaunay triangulations this was similar. So, the quad-edge structure may not be a good idea.

INSERT YOUR IMAGE HERE!

Figure 4.10: ***Polyhedral Surface Data Structure*** *Scheme and Memory Layout of triangles and vertices*

4.3 Surface Mesh Preprocessing

The goal of preprocessing a surface mesh which has been loaded from disk is to transform its data into a valid and efficient representation of a polyhedral surface that should exhibit functionality to draw and move points and discrete curves on its surface. In the case that this transformation is not possible, the stage at least must check for the validity of the given data and emit an error if there might be any violations.

A given Mesh for the algorithm can be quite general. Providing triangles which will introduce numerical difficulties is out of the scope of this thesis.

We will focus on orientable meshes. Please note, that is not an actual restriction for the algorithm but will only speed up the implementation. Furthermore, we look at surfaces which are the boundary of volumes in three-dimensional Euclidean space which can be looked at as open submanifolds. These volumes are therefore by definition oriented and, hence, their boundary needs to be, too. Still, we need to look boundaries for surfaces as, for example, distance envelopes restrict the surface without boundary to be one. The boundaries need to be handled properly. Also, these requirements are only need to hold for the restriction to the distance envelope. Often, lines will be drawn on oriented/orientable parts of the surface even if the surface is not orientable due to artifacts originating from scanning or generating meshes.

Topological Connections We need the topological connections of a given surface mesh. In the case of general file formats, a scene is typically separated into multiple meshes which are topologically but not smoothly connected. For proper drawing of curves in a whole scene, the topological structure of connections needs to be generated first. Using the STL file format, this is not needed.

INSERT YOUR IMAGE HERE!

Figure 4.11: ***Examples of Polyhedral Surfaces*** *The images shows oriented polyhedral surfaces with and without boundaries, also called models.*

Generating the dual graph To generate such a dual graph we refer to this book. It is assumed to be a solved problem. First, we have created a hash map of oriented edges with a custom hash function to mangle vertex indices. The mapped values store the faces indices and their location in the triangle. By inverting the indices of a stored edge, one can easily access the neighboring triangle if it exists. Furthermore, it is easy to check whether the mesh is oriented and a valid two-dimensional manifold.

Check for orientability and validity

4.4 Discrete Surface Curve Data Structure

There two main attempts for the construction of a data structure representing surface mesh curves. One of them is based on edges and the other on triangles.

We do not look at polyhedral surfaces as an approximation of a real-world smooth manifold.

Advantages of discrete surface mesh curves over more general curves on polyhedral surfaces

- Curve is provided in triangle precision. Better Compression. More points would be waste of memory.
- Robust: Snap to vertex if points get arbitrary near to each other
- curve is always a surface curve of the polyhedral surface and not violating mathematical properties. This also allows for direct rendering without projection.
- Useful for mesh processing
- Interpretation of Discretization is more consistent: surface mesh may be discretized approximationof real-world manifold with the mesh to be the finite grid. then it is only natural to discretize a real world surface curve in the same way.

Advantages of the triangular data structure over the edge-based one:

- Triangles can be easier generalized to higher dimensions than edges.
- Triangles do not need to handle boundaries in complicated ways.
- Triangles allow easy and ordered access to fan around vertex.
- Artifact removal routine for triangles is simpler than for edges.
- Triangles do not exhibit data loss when handling reflex vertices
- Triangles allow for a more general moving of points

4.5 Initial Curve Selection

As for the vast majority of optimization algorithms, the results and efficient working are highly dependent on the chosen starting values. Curve smoothing itself is an optimization process and to solve the boundary value problem, we need an initial value. So, choosing the initial curve in the right way will also heavily change the speed and quality of the algorithm. The algorithm should be able to handle a vast amount unsmooth curves. Still, the handling of artifacts will be taken care of at the start.

User Interface for Selecting and Controlling Initial Curves

Drawing by Ray Tracing

Connecting the Vertices

Closed Initial Curves and Fixed Vertices

Artifact Removal

Smoothed Curvature Values by Stencil Stencil is discrete approximation of solution to Laplacian equation or heat equation which is smooth (infinitely differentiable). Maybe it would be useful to keep the length of the curve and still smooth it.

Vertex Curves

Face Curves

Tracing Geodesics

4.6 Unfolding

To unfold two triangles along their common edge and find the shortest line connecting two points, we will first look back into two dimensions.

INSERT YOUR IMAGE HERE!

Figure 4.12: ***2D Unfolding Primitive***

Let $p, q \in \mathbb{R}^2$ such that $p_x < q_x$.

$$\Delta x = q_x - p_x \qquad \Delta y = q_y - p_y$$

The straight line function connecting these two points then looks like the following.

$$f(x) = \frac{q_y - p_y}{q_x - p_x}(x - p_x) + p_y = \frac{q_y - p_y}{q_x - p_x}x + \frac{p_y q_x - q_y p_x}{q_x - p_x}$$

Please note that this function is well-defined, as $q_x - p_x > 0$. To get the intersection point with the ordinate, set the argument to zero.

$$t \coloneqq f(0) = \frac{p_y q_x - q_y p_x}{q_x - p_x}$$

INSERT YOUR IMAGE HERE!

Figure 4.13: ***Unfolding of two Triangles***

Applying this expression to the case of two triangles that needs to be unfolded, the following results.

$$e \coloneqq \frac{v_2 - v_1}{\|v_2 - v_1\|} \qquad p \coloneqq r_1 - v_1 \qquad q \coloneqq r_2 - v_1$$

$$p_y = \langle e \,|\, p \rangle \qquad q_y = \langle e \,|\, q \rangle$$

$$p_x = -\|p - p_y e\| \qquad q_x = \|q - q_y e\|$$

Unfolding two adjacent triangles that share a common edge from three-dimensional space into the two-dimensional plane is not unique as there are two cases. In one case, the triangles might overlap and in the other not. The above sign for p_x is therefore important.

> **LEMMA 4.1:** Unfolding leads to geodesic
>
> *Connecting p, r, and q in that order by straight lines leads to locally shortest and straightest geodesics. It solves the discrete boundary value problem for locally shortest and straightest geodesics when boundary points lie in the inside of adjacent triangles.*

PROOF:
According to Polthier and Schmies (2006) and Martínez, Velho, and Carvalho (2005), locally shortest and straightest geodesics coincide on polyhedral if they do not contain any vertices. So, it is sufficient to show that the discrete curvature at the crease is zero. But this has be true by construction. □

4.7 Unfolding with Desired Curvature

Again, a two-dimensional primitive is used.

$$f(x) = -\cot\alpha(x - p_x) + p_y$$

$$t = f(0) = p_y + |p_x| \cot\alpha$$

Now, combine two points to both sides with a desired discrete curvature. For this, assume $p_x < 0$ and $q_x > 0$. Furthermore, the desired curvature is positive $\kappa > 0$ and taken mathematically positive.

$$\kappa + \pi = \alpha + \beta$$

From the former computation, it is clear that their points on the ordinate have to coincide.

$$p_y - p_x \cot\alpha = q_y + q_x \cot\beta$$

$$\cot\beta = \cot(\pi - (\alpha - \kappa)) = -\cot(\alpha - \kappa)$$

$$\cot(\alpha - \kappa) = \frac{\cot\alpha\cot\kappa + 1}{\cot\alpha - \cot\kappa}$$

$$p_y - p_x \cot\alpha = q_y - q_x \frac{\cot\alpha\cot\kappa + 1}{\cot\alpha - \cot\kappa}$$

$$\varphi \coloneqq \cot\alpha \qquad K \coloneqq \cot\kappa$$

$$p_y - p_x\varphi = q_y - q_x \frac{K\varphi + 1}{\varphi - K}$$

$$q_x \frac{K\varphi + 1}{\varphi - K} = q_y - p_y + p_x\varphi$$

$$Kq_x\varphi + q_x = (q_y - p_y)(\varphi - K) + p_x\varphi(\varphi - K)$$

$$Kq_x\varphi + q_x = \Delta y\varphi - K\Delta y + p_x\varphi^2 - Kp_x\varphi$$

$$q_x + K\Delta y = p_x\varphi^2 + (\Delta y - K(p_x + q_x))\varphi$$

$$\frac{q_x + K\Delta y}{p_x} = \varphi^2 + \frac{\Delta y - K\Sigma x}{p_x}\varphi$$

$$\varphi = -\frac{\Delta y - K\Sigma x}{2p_x} \pm \sqrt{\frac{q_x + K\Delta y}{p_x} + \frac{(\Delta y - K\Sigma x)^2}{4p_x^2}}$$

$$t = p_y - p_x\varphi$$

$$t = p_y + \frac{\Delta y - K\Sigma x}{2} \mp \sqrt{p_x(q_x + K\Delta y) + \frac{(\Delta y - K\Sigma x)^2}{4}}$$

$$t = \frac{\Sigma y - K\Sigma x}{2} \mp \sqrt{p_x(q_x + K\Delta y) + \frac{\Delta^2 y - 2K\Delta y\Sigma x + K^2\Sigma^2 x}{4}}$$

$$t = \frac{\Sigma y - K\Sigma x}{2} \mp \sqrt{p_x q_x(1 + K^2) + \frac{\Delta^2 y - 2K\Delta y\Delta x + K^2\Delta^2 x}{4}}$$

$$t = \frac{\Sigma y - K\Sigma x}{2} \mp \sqrt{p_x q_x(1 + K^2) + \frac{(\Delta y - K\Delta x)^2}{4}}$$

4.8 Desired Curvature Stencil

4.9 Desired Curvature Mapping

4.10 Curve Smoothing Algorithm

Curve smoothing is not a discrete problem. Using only a finite amount of steps is probably not possible. So, we use a process of convergence. This has the advantage that not all triangles need to be unfolded at once. This makes the implementation and parallelization much simpler. The discrete algorithms for tracing geodesics are not easy to parallelize.

Idea and Overview

Edge Vertex Relaxation We need to take a look at topological and numerical robustness.

Vertex Vertex Relaxation

Critical Vertex Handling

Artifact Removal and Self-Intersection Handling

Desired Curvature Mapping Desired curvatures need to be constant along edges. Therefore interpolate on angle-basis around vertex. We do not want to loose information of desired curvatures.

Curve Evaluation

Correctness and Convergence Correctness can be shown by showing the convergence to the curvature values. This by definition of the given curvature values smooths the curve. The convergence might only be shown for contracting the curve by smoothing. As the limit is the geodesic, the prove of its convergence is there.

5 Implementation

The implementation will explain the implementation of the curve smoothing and not the of other stages of the pipeline. Maybe geodesics tracing.

5.1 Serial Implementation on CPU

Data Structure for Curve Vertices The stored edge is oriented.

Edge Vertex Relaxation

Vertex Vertex Relaxation

Critical Vertices

Loop

5.2 Parallelization and Vectorization

Vectorization takes place at a low level while parallelization takes place at a high level. Using SIMD vectorization, one needs to make sure that the registers are full. For Edge Vertices, this is no problem. For Vertex Vertex Handling, this might not increase the handling at all. To let the compiler automatically vectorize wherever possible, divide edge vertices and vertex vertices and different sets. Is this still useful?

Vectorization might not be the source for efficiency of this algorithm and it cannot be applied directly to GPU programming. Parallelization will most likely need to take care of neighboring points. Stream-based serial process needs to be changed.

5.3 Parallel Implementation on the CPU

5.4 Implementation on the GPU

6 Testing

6.1 Two-Dimensional Case

For all cases, we test the smoothing and the geodesics generation. How many iterations for convergence?

Regular Grids

Irregular Grids This may also be called parametric noise.

Grids with Inner or Non-Convex Boundaries

Lines with Self-Intersection

Closed Curves

6.2 Three-Dimensional Case

Small Artificial Tests

Closed Curves

Tests on Meshes

Meshes with Geometric and Parametric Noise

7 Application

8 Evaluation and Results

9 Conclusions and Future Work

References

Books

Carmo, Manfredo P. Do (2016). *Differential Geometry of Curves and Surfaces*. Revised & Updated Second Edition. Dover Publications. ISBN: 978-0-486-80699-0.

Cheney, Ward and David Kincaid (2008). *Numerical Mathematics and Computing*. Sixth Edition. Thomson Brooks/Cole. ISBN: 978-0-495-11475-8.

Elstrodt, Jürgen (2018). *Maß- und Integrationstheorie*. Eighth Edition. Springer Spektrum. ISBN: 978-3-662-57938-1. DOI: 10.1007/978-3-662-57939-8.

Forster, Otto (2016). *Analysis 1. Differential- und Integralrechnung einer Veränderlichen*. Springer Spektrum. ISBN: 978-3-658-11544-9. DOI: 10.1007/978-3-658-11545-6.

Goldhorn, Karl-Heinz, Hans-Peter Heinz, and Magarita Kraus (2009). *Moderne mathematische Methoden der Physik*. Vol. 1. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-88543-6. DOI: 10.1007/978-3-540-88544-3.

Kühnel, Wolfgang (2013). *Differentialgeometrie*. Sixth Edition. Springer Spektrum. ISBN: 978-3-658-00614-3. DOI: 10.1007/978-3-658-00615-0.

McCool, Michael, Arch D. Robison, and James Reinders (2012). *Structured Parallel Programming: Patterns of Efficient Computation*. Morgan Kaufmann – Elsevier. ISBN: 978-0-12-415993-8.

Meyers, Scott (2014). *Effective Modern C++*. O'Reilly Media. ISBN: 978-1-491-90399-5.

Munzner, Tamara (2014). *Visualization Analysis and Design*. A K Peters Visualization Series. CRC Press. URL: https://www.cs.ubc.ca/~tmm/vadbook/ (visited on 11/16/2022).

Patterson, David A. and John L. Hennessy (2014). *Computer Organization and Design. The Hardware/Software Interface*. Fifth Edition. Morgan Kaufmann – Elsevier. ISBN: 978-0-12-407726-3.

Reddy, Martin (2011). *API Design for C++*. Morgan Kaufmann – Elsevier. ISBN: 978-0-12-385003-4.

Stahl, Saul and Catherine Stenson (2013). *Introduction to Topology and Geometry*. Second Edition. John Wiley & Sons. ISBN: 978-1-118-10810-9.

Stroustrup, Bjarne (2014). *The C++ Programming Language*. Fourth Edition. Addison-Wesley – Pearson Education. ISBN: 978-0-321-95832-7.

Vandevoorde, David, Nicolai M. Josuttis, and Douglas Gregor (2018). *C++ Templates: The Complete Guide*. Second Edition. Addison-Wesley – Pearson Education. ISBN: 978-0-321-71412-1.

Williams, Anthony (2019). *C++ Concurrency in Action*. Second Edition. Manning Publications. ISBN: 978-1-61-729469-3.

Online Resources

cppreference.com (n.d.). URL: https://en.cppreference.com/w/ (visited on 01/15/2023).

MathWorld, Wolfram (2022). *Teardrop Curve*. URL: https://mathworld.wolfram.com/TeardropCurve.html (visited on 11/28/2022).

OpenGL: The Industry's Foundation for High Performance Graphics (2023). URL: https://www.opengl.org/ (visited on 01/15/2023).

Pawellek, Markus (2023). *reflex. Reactive and Flexible Curve Smoothing on Surface Meshes*. URL: https://github.com/lyrahgames/reflex (visited on 01/15/2023).

Standard C++ Foundation (2023). URL: https://isocpp.org/ (visited on 01/15/2023).

Yocto/GL (2023). *Tiny C++ Libraries for Data-Oriented Physically-based Graphics*. URL: https://github.com/xelatihy/yocto-gl (visited on 11/19/2022).

General

Alirr, Omar and Ashrani Aizzuddin Abd. Rahni (August 2019). "Survey on Liver Tumour Resection Planning System: Steps, Techniques, and Parameters". In: *Journal of Digital Imaging* 33. DOI: 10.1007/s10278-019-00262-8.

Benhabiles, Halim et al. (December 2011). "Learning Boundary Edges for 3D-Mesh Segmentation". In: *Computer Graphics Forum* 30, pp. 2170–2182. DOI: 10.1111/j.1467-8659.2011.01967.x.

Bischoff, Stephan, Tobias Weyand, and Leif Kobbelt (January 2005). "Snakes on Triangle Meshes". In: *Proceedings of Bildverarbeitung für die Medizin*, pp. 208–212. DOI: 10.1007/3-540-26431-0_43.

Bommes, David and Leif Kobbelt (January 2007). "Accurate Computation of Geodesic Distance Fields for Polygonal Curves on Triangle Meshes". In: *Proceedings of the Vision, Modeling, and Visualization Conference*, pp. 151–160. URL: https://www-sop.inria.fr/members/David.Bommes/publications/geodesic.pdf (visited on 11/16/2022).

Chaikin, George (December 1974). "An Algorithm for High-Speed Curve Generation". In: *Computer Graphics and Image Processing* 3, pp. 346–349. DOI: 10.1016/0146-664X(74)90028-8.

Crane, Keenan, Clarisse Weischedel, and Max Wardetzky (September 2013). "Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow". In: *ACM Transactions on Graphics* 32. DOI: 10.1145/2516971.2516977.

Crane, Keenan et al. (2020). *A Survey of Algorithms for Geodesic Paths and Distances*. DOI: 10.48550/ARXIV.2007.10430. URL: https://arxiv.org/abs/2007.10430 (visited on 02/18/2023).

Dijkstra, Edsger W. (1959). "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1, pp. 269–271. URL: https://www.semanticscholar.org/paper/A-note-on-two-problems-in-connexion-with-graphs-Dijkstra/45786063578e814444b8247028970758bbbd0488 (visited on 11/17/2022).

Dyn, Nira, D. Levin, and D. Liu (April 1992). "Interpolatory Convexity-Preserving Subdivision Schemes for Curves and Surfaces". In: *Computer-Aided Design* 24, pp. 211–216. DOI: 10.1016/0010-4485(92)90057-H.

Engelke, Wito et al. (August 2018). "Autonomous Particles for Interactive Flow Visualization". In: *Computer Graphics Forum* 38. DOI: 10.1111/cgf.13528.

Guibas, Leonidas and Jorge Stolfi (April 1985). "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams". In: *ACM Transactions on Graphics* 4, pp. 74–123. DOI: 10.1145/282918.282923. URL: http://sccg.sk/~samuelcik/dgs/quad_edge.pdf (visited on 11/07/2020).

Hertzmann, Aaron and Denis Zorin (2000). "Illustrating Smooth Surfaces". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIG-

GRAPH '00. ACM Press/Addison-Wesley Publishing Co., 517–526. ISBN: 1581132085. DOI: 10.1145/344779.345074.

Hofer, Michael and Helmut Pottmann (August 2004). "Energy-Minimizing Splines in Manifolds". In: *ACM Transactions on Graphics* 23, pp. 284–293. DOI: 10.1145/1015706.1015716.

Ji, Zhongping et al. (September 2006). "Easy Mesh Cutting". In: *Computer Graphics Forum* 25, pp. 283–291. DOI: 10.1111/j.1467-8659.2006.00947.x.

Jin, Shuangshuang, Robert Lewis, and David West (February 2005). "A Comparison of Algorithms for Vertex Normal Computation". In: *The Visual Computer* 21, pp. 71–82. DOI: 10.1007/s00371-004-0271-1.

Jung, Moonryul and Haengkang Kim (November 2004). "Snaking Across 3D Meshes". In: *Proceedings of Pacific Graphics*, pp. 87–93. DOI: 10.1109/PCCGA.2004.1348338.

Kaplansky, Lotan and Ayellet Tal (October 2009). "Mesh Segmentation Refinement". In: *Computer Graphics Forum* 28, pp. 1995–2003. DOI: 10.1111/j.1467-8659.2009.01578.x.

Kass, Michael, Andrew Witkin, and Demetri Terzopoulos (January 1988). "Snakes: Active Contour Models". In: *IEEE Proceedings on Computer Vision and Pattern Recognition* 1, pp. 321–331. URL: https://sites.pitt.edu/~sjh95/related_papers/Kass1988_Article_SnakesActiveContourModels.pdf (visited on 11/16/2022).

Kimmel, Ron and J. A. Sethian (1996). *Fast Marching Methods for Computing Distance Maps and Shortest Paths*. Tech. rep. Lawrence Berkeley National Laboratory. URL: https://escholarship.org/uc/item/7kx079v5 (visited on 11/16/2022).

— (August 1998). "Computing Geodesic Paths on Manifolds". In: *Proceedings of the National Academy of Sciences of the United States of America* 95, pp. 8431–5. DOI: 10.1073/pnas.95.15.8431.

Kindlmann, Gordon et al. (November 2003). "Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications". In: vol. 2003, pp. 513–520. ISBN: 0-7803-8120-3. DOI: 10.1109/VISUAL.2003.1250414.

Lai, Yu-Kun et al. (January 2007). "Robust Feature Classification and Editing". In: *IEEE Transactions on Visualization and Computer Graphics* 13, pp. 34–45. DOI: 10.1109/TVCG.2007.19.

Lawonn, Kai et al. (2014). "Adaptive and Robust Curve Smoothing on Surface Meshes". In: *Computers & Graphics* 40, pp. 22–35. DOI: 10.1016/j.cag.2014.01.004.

Lee, D. and F. Preparata (September 1984). "Euclidean Shortest Paths in the Presence of Rectilinear Barriers". In: *Networks* 14, pp. 393–410. DOI: 10.1002/net.3230140304.

Lee, Yunjin and S. Lee (September 2002). "Geometric Snakes for Triangular Meshes". In: *Computer Graphics Forum* 21, pp. 229 –238. DOI: 10.1111/1467-8659.t01-1-00582.

Lee, Yunjin et al. (January 2004). "Intelligent Mesh Scissoring Using 3D Snakes". In: pp. 279–287. DOI: 10.1109/PCCGA.2004.1348358.

Lévy, Bruno et al. (July 2002). "Least Squares Conformal Maps for Automatic Texture Atlas Generation". In: *ACM Transactions on Graphics* 21, pp. 362–371. DOI: 10.1145/566654.566590.

Ma, Li and Dezhong Chen (June 2007). "Curve Shortening in a Riemannian Manifold". In: *Annali Di Matematica Pura Ed Applicata* 186, pp. 663–684. DOI: 10.1007/s10231-006-0025-y.

Mancinelli, Claudio et al. (May 2022). "b/Surf: Interactive Bzier Splines on Surface Meshes". In: *IEEE Transactions on Visualization and Computer Graphics* PP. DOI: 10.1109/TVCG.2022.3171179.

Martínez, Dimas, Paulo de Carvalho, and Luiz Velho (November 2007). "Geodesic Bezier Curves: A Tool for Modeling on Triangulations". In: *Brazilian Symposium on Computer Graphics and Image Processing*, pp. 71–78. ISBN: 978-0-7695-2996-7. DOI: 10.1109/SIBGRAPI.2007.38.

Martínez, Dimas, Luiz Velho, and Paulo de Carvalho (October 2005). "Computing Geodesics on Triangular Meshes". In: *Computers & Graphics* 29, pp. 667–675. DOI: 10.1016/j.cag.2005.08.003.

Max, Nelson (January 1999). "Weights for Computing Vertex Normals from Facet Normals". In: *Journal of Graphics Tools* 4. DOI: 10.1080/10867651.1999.10487501.

Meyer, Mark et al. (November 2001). "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds". In: *Proceedings of Visualization and Mathematics* 3. DOI: 10.1007/978-3-662-05105-4_2.

Mitchell, Joseph, David Mount, and Christos Papadimitriou (August 1987). "The Discrete Geodesic Problem". In: *SIAM Journal on Computing* 16, pp. 647–668. DOI: 10.1137/0216045.

Morera, Dimas Martínez, Luiz Velho, and Paulo Cezar Pinto Carvalho (2008). "Subdivision Curves on Triangular Meshes". In: URL: https://www.semanticscholar.org/paper/Subdivision-Curves-on-Triangular-Meshes-Morera-Velho/595d28aacea33ba038d36e7dc403c156a9248905 (visited on 11/16/2022).

Park, Jongha et al. (May 2019). "Fully Automated Lung Lobe Segmentation in Volumetric Chest CT with 3D U-Net: Validation with Intra- and Extra-Datasets". In: *Journal of Digital Imaging* 33. DOI: 10.1007/s10278-019-00223-1.

Pellacini, Fabio, Giacomo Nazzaro, and Edoardo Carra (2019). "Yocto/GL: A Data-Oriented Library For Physically-Based Graphics". In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. Ed. by Marco Agus, Massimiliano Corsini, and Ruggero Pintus. The Eurographics Association. ISBN: 978-3-03868-100-7. DOI: 10.2312/stag.20191373.

Polthier, Konrad and Markus Schmies (2006). "Straightest Geodesics on Polyhedral Surfaces". In: *ACM SIGGRAPH 2006 Courses*. SIGGRAPH '06. Association for Computing Machinery, 30–38. DOI: 10.1145/1185657.1185664.

Pottmann, Helmut and Michael Hofer (October 2005). "A Variational Approach to Spline Curves on Surface". In: *Computer Aided Geometric Design* 22, pp. 693–709. DOI: 10.1016/j.cagd.2005.06.006.

Rusinkiewicz, Szymon (October 2004). "Estimating Curvatures and Their Derivatives on Triangle Meshes". In: pp. 486–493. ISBN: 0-7695-2223-8. DOI: 10.1109/TDPVT.2004.1335277.

Sethian, J. A. (March 1996). "A Marching Level Set Method for Monotonically Advancing Fronts". In: *Proceedings of the National Academy of Sciences of the United States of America* 93, pp. 1591–5. DOI: 10.1073/pnas.93.4.1591.

Sharp, Nicholas and Keenan Crane (December 2020). "You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges". In: *ACM Transactions on Graphics* 39.6, pp. 1–15. DOI: 10.1145/3414685.3417839.

Surazhsky, Vitaly et al. (July 2005). “Fast Exact and Approximate Geodesics on Meshes”. In: *ACM Transactions on Graphics* 24, pp. 553–560. DOI: 10.1145/1073204.1073228.

Xin, Shi-Qing and Guo-Jin Wang (December 2007). “Efficiently Determining a Locally Exact Shortest Path on Polyhedral Surfaces”. In: *Computer-Aided Design* 39, pp. 1081–1090. DOI: 10.1016/j.cad.2007.08.001.

Yu, Chris, Henrik Schumacher, and Keenan Crane (April 2021). “Repulsive Curves”. In: *ACM Transactions on Graphics* 40, pp. 1–21. DOI: 10.1145/3439429.

Zachow, Stefan et al. (2003). “Draw and Cut: Intuitive 3D Osteotomy Planning on Polygonal Bone Models”. In: *International Congress Series* 1256, pp. 362–369. DOI: 10.1016/S0531-5131(03)00272-3.

A Analysis on Manifolds

Topological manifolds are the basis of differentiable manifolds and describe the class polyhedral surfaces lie in.

DEFINITION A.1: Topological Manifold

Let $n \in \mathbb{N}$. Then a topological n-dimensional manifold M (with boundary) is a second countable Hausdorff space which is locally homeomorphic to $\mathbb{H}^n$.
That is, for all $p \in M$, there exists an open neighborhood U of p in M, an open set $V \subset \mathbb{H}^n$, and a homeomorphism $\varphi\colon U \to V$, called a (coordinate) chart.

For the purpose of this thesis, all spaces will be a second countable Hausdorff space. The definition is given for completeness. A topological manifold with boundary is generalization of the standard concept of topological manifolds without boundary. A topological manifold without boundary is a topological manifold with boundary, whereby its boundary is given by the empty set.

DEFINITION A.2: Smooth Manifold

Let $n \in \mathbb{N}$, $k \in \mathbb{N}_\infty$, and M be a topological n-dimensional manifold. Then, given two charts (U, φ) and (V, ψ) of M, their transition map, also known as coordinate transformation, is defined by the following composition.

$$\varphi|_{U\cap V} \circ \psi|_{U\cap V}^{-1}\colon \psi(U \cap V) \to \varphi(U \cap V)$$

The charts (U, φ) and (V, ψ) are said to be C^k-compatible if either $U \cap V = \emptyset$ or their transition map is a C^k-diffeomorphism.
A C^k-atlas of M is a family of C^k-compatible charts that covers all of M.
By equipping the topological manifold M with a maximal C^k-atlas, we obtain a differentiable n-dimensional manifold of class C^k (with boundary).
For $k = \infty$, it is called a smooth n-dimensional manifold (with boundary).

DEFINITION A.3: Smooth Maps

Let M and N be two manifolds and $F\colon M \to N$. For two given charts (U, φ) of M and (V, ψ) of N, we define the coordinate representation of F by the following composition.

$$\psi \circ F \circ \varphi|_{U\cap F^{-1}(V)}^{-1}\colon \varphi\left(U \cap F^{-1}(V)\right) \to \psi(V)$$

F is a differentiable map of class C^k, if for all pairs of given charts, its coordinate representations is an element of $\mathrm{C}^k(\mathbb{R}^m, \mathbb{R}^n)$.
We call F a C^k-diffeomorphism if it is bijective and in both directions a differentiable map of class C^k.

$$\mathrm{C}^k(M, N) \coloneqq \left\{F\colon M \to N \;\middle|\; F \text{ is differentiable of class } \mathrm{C}^k\right\}$$

$$\mathrm{C}^k(M) \coloneqq \mathrm{C}^k(M, \mathbb{R})$$

DEFINITION A.4: Tangential Space

Let M be a smooth manifold and $p \in M$. A tangential vector in p is a linear and smooth functional $X\colon \mathrm{C}^\infty(M) \to \mathbb{R}$, such that for all $f, g \in \mathrm{C}^\infty(M)$ the following holds.

$$X(fg) = f(p)X(g) + g(p)X(f)$$

The set T_pM of all tangential vectors in p is called tangential space in p. The tangent bundle is then as disjoint union.

$$\mathrm{T}M \coloneqq \bigsqcup_{p \in M} \mathrm{T}_pM$$

The tangential space is for all points isomorphic to $\mathbb{R}^n$. For a chosen chart, we can easily construct basis vectors. Every tangential vector is a tangent vector of a curve running through that point.

DEFINITION A.5: Differential

$$\mathrm{d}F(p)\colon \mathrm{T}_pM \to \mathrm{T}_{F(p)}N \qquad \mathrm{d}F(p)(X)(f) \coloneqq X(f \circ F)$$
$$\mathrm{d}F\colon \mathrm{T}M \to \mathrm{T}N$$

DEFINITION A.6: Smooth Embedding

Let M and N be two manifolds and $F\colon M \to N$. F is an immersion if for all $p \in M$ its differential $\mathrm{d}F(p)$ is injective. Furthermore, it is called an embedding if $F\colon M \to F(M)$ is a homeomorphism.

DEFINITION A.7: Embedded Submanifold

Let M be a manifold. Then a subset $S \subset M$ is called an embedded submanifold of dimension k if for all $p \in S$, there exists a chart (U, φ) in M with $p \in U$, such that the following holds.

$$\varphi(U \cap S) = \{x \in \varphi(U) \mid \forall p \in \mathbb{N}, k < p \leq n\colon x_p = 0\}$$

THEOREM A.1: The image of an embedding is an embedded submanifold

The image of an embedding is an embedded submanifold.

DEFINITION A.8: Vector Bundle

Let M be a manifold. Then a vector bundle E rank k over M is a manifold equipped with differentiable and surjective projection $\pi\colon E \to M$, such that the

following properties hold.

- *For all $p \in M$, the fiber $\pi^{-1}(p) \subset E$ is isomorphic to $\mathbb{R}^k$.*
- *locally trivial: For all $p \in M$, there is a neighborhood U of p in M and a diffeomorphism $\varphi\colon \pi^{-1}(U) \to U \times \mathbb{R}^k$, such that $\pi = \pi_1 \circ \varphi$ and the restriction of φ to the fibers $\pi^{-1}(p)$ is a linear isomorphism to $\mathbb{R}^k$.*

A map $\sigma\colon M \to E$ with $\pi \circ \sigma = \mathrm{id}_M$ is called a section.

DEFINITION A.9: Tensorbundle

For a finite dimensional real vector space V, a k-tensor is a multilinear functional $T\colon V^k \to \mathbb{R}$. The space of all k-tensors over V is denoted by $\mathrm{T}^k(V)$. For a manifold M, we define the k-tensor bundle as follows.

$$\mathrm{T}^k M \coloneqq \bigsqcup_{p \in M} \mathrm{T}^k(\mathrm{T}_p M)$$

DEFINITION A.10: Riemannian Manifold

A Riemannian manifold is a smooth manifold M equipped with a positive-definite inner product g_p on the tangent space $\mathrm{T}_p M$ at each point $p \in M$. Thereby, for any chosen chart the coordinate functions of g need to be smooth.

The inner metric of a Riemannian manifold takes the place of the scalar product and measures angles.

THEOREM A.2: For every smooth manifold, there exists a Riemannian metric

For every smooth manifold, there exists a Riemannian metric.

DEFINITION A.11: Normal Space

Let (M, g) be a Riemannian manifold, $S \subset M$ a Riemannian submanifold and $p \in S$.

$$\mathrm{N}_p S \coloneqq \{x \in \mathrm{T}_p M \mid \forall y \in \mathrm{T}_p S\colon g(x, y) = 0\}$$

The normal space is the orthogonal complement and can in general only be defined for embedded manifolds. For orthogonality, we need a Riemannian manifold. A more general definition is possible by using quotient spaces. As a consequence, the normal space is isomorphic to $\mathbb{R}^{n-k}$.

B Mathematical Proofs

C Further Code

Statutory Declaration

I declare that I have developed and written the enclosed Master's thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master's thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.
On the part of the author, there are no objections to the provision of this Master's thesis for public use.

Bergen, February 22, 2023

Markus Pawellek