

Technische Universität München

TUM School of Computation, Information and Technology



Parallelized Approaches to Conformal Mappings with applications to real-time calligraphy

Lena Polke

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung einer

Doktorin der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Christina Kuttler

Prüfer der Dissertation:

1. Prof. Dr. Dr. Jürgen Richter-Gebert
2. Prof. Dr. Craig S. Kaplan
3. Prof. Dr. Martin Skrodzki

Die Dissertation wurde am 11.12.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 08.05.2024 angenommen.

Acknowledgment

I am very grateful to know so many supportive and inspiring people who have helped me to finish this thesis.

First of all, I want to thank my supervisor, Jürgen Richter-Gebert, for giving me the opportunity to achieve more than I ever thought I could. I also thank you for believing in my ability to teach - I have learned to like it very much. Also, I would like to thank Craig Kaplan and Martin Skrodzki for reviewing my dissertation and Christina Kuttler for chairing the committee.

Furthermore, I would like to thank everyone who took the time to proofread parts of this thesis: Ati, Bernhard, Franzi, Isa, Jannik, Kilian, Martin, Quirin, Stefan and Tim. Your comments and suggestions were incredibly helpful.

I also thank all of my colleagues for the nice cookie breaks, lunch talks, Enigames and seminars. Special thanks go to Jutta, who was my first contact in the working group and who has helped me with many questions since then. Thank you, Aaron, for having been a wonderful roommate and for introducing me to the art of implementation. I am also very grateful to Bernhard for discussing teaching concepts and showing me how to answer difficult questions from students. Thank you, Tim, for being a great colleague and doing last-minute proofreading.

A big thank you goes to my friends for listening to both enthusiastic reports and complaints about my work. Without the Tuesday nights, the singing, the shared dinners, the game evenings, the movie nights, the hiking, and the dancing, I would not have been able to come up with what is part of this thesis. I especially want to thank Franzi for sharing your experience as a doctoral candidate, your knowledge of how to write a dissertation, and for being a great role model. Thank you, Isa, for reassuring me that revising texts actually makes them better.

Mama, thank you for everything, especially for not giving me apples and for asking the right questions at the right time. Thank you, Ati, for comments like “What are you doing in your thesis besides drawing?” and for being you, you’re the best. Last but not least, thank you, Michael, for always being there for me and for believing that I could actually do this.

Abstract

In this thesis, we propose an algorithm that approximates a conformal, i.e., an angle and infinitesimal shape preserving map of a rectangular strip to a user-defined stroke domain. The rectangular preimage domain consists of ornamental tiles. The stroke domain is defined by a pressure-sensitive digital pen model based on a user-drawn path on a digital drawing surface. We investigate the underlying continuous model in detail, which is essential for comprehending the structure of the target domain of the conformal map. In particular, we use techniques from differential geometry and catastrophe theory to identify and classify singular points on the boundary of a stroke. One of the main contributions of this thesis is the ability to handle constantly changing target domains of a conformal map: our algorithm computes the conformal map of the preimage domain to the stroke in real time during the drawing process. Due to the extensive study of a stroke's boundary, our algorithm also behaves well at the boundary of the map's target domain. Furthermore, we are capable of dealing with self-intersecting strokes. To ensure that our computations are fast enough to keep up with the drawing process, we implement our algorithm for the graphics processing unit (GPU). Computations on the GPU are executed on graphical textures composed of pixels, on which highly parallel calculations are performed. We ensure conformality of the map by iteratively applying pixel averaging to the preimage coordinates for each pixel within the stroke. As the result of each iteration is displayed, the artistic content of the preimage visually converges to its intended conformal position as the stroke is drawn. We provide both local and global tests for the accuracy of our algorithm by applying results from continuous and discrete theory about conformality. Those tests indicate that the outcome of our map is nearly conformal. We extend our algorithm to allow variable design based on rules acting on the conformally mapped coordinate system in the stroke.

Zusammenfassung

In dieser Arbeit stellen wir einen Algorithmus vor, der eine konforme, d.h. eine winkeltreue und infinitesimale Formen erhaltende Abbildung eines rechteckigen Musters auf einen benutzerdefinierten Stiftstrich näherungsweise berechnet. Dieser Strich wird durch ein druckempfindliches digitales Stiftmodell definiert, das auf einer von Hand gezeichneten Kurve auf einer digitalen Zeichenoberfläche basiert. Wir erstellen eine detaillierte Analyse des zugrundeliegenden kontinuierlichen Modells, was unerlässlich dafür ist, die Struktur des Bildbereichs der konformen Abbildung zu verstehen. Insbesondere verwenden wir Techniken aus der Differentialgeometrie und der Katastrophentheorie, um singuläre Punkte am Rande eines Strichs zu identifizieren und zu klassifizieren. Einer der entscheidenden Beiträge dieser Arbeit ist die Fähigkeit, mit sich ständig ändernden Bildbereichen einer konformen Abbildung umzugehen: unser Algorithmus berechnet die annähernd konforme Abbildung des Musters auf den Strich in Echtzeit während des Zeichenvorgangs. Dank der umfassenden Betrachtung der Randkurven unseres Modells ist die berechnete Abbildung auch am Rand eines Strichs näherungsweise konform. Darüber hinaus kann unser Algorithmus auch mit sich selbst überschneidenden Strichen umgehen. Um sicherzustellen, dass unsere Berechnungen schnell genug sind, um mit dem Zeichenprozess Schritt zu halten, implementieren wir unseren Algorithmus für die Grafikkarte (GPU). Die Berechnungen auf der GPU werden auf grafischen Texturen ausgeführt, die aus Pixeln bestehen, auf denen hochparallele Berechnungen durchgeführt werden. Wir nutzen iterativ angewandte Durchschnittsberechnung auf den Koordinaten der einzelnen Pixel im Stiftstrich, um die Abbildung zu berechnen. Da das Ergebnis jeder Iteration im Muster des Strichs angezeigt wird, konvergiert der künstlerische Inhalt visuell sichtbar zu seiner beabsichtigten konformen Position, während der Strich gezeichnet wird. Anhand von Resultaten über kontinuierliche und diskrete konforme Abbildungen testen wir die Konformität unseres Algorithmus sowohl lokal als auch global. Diese Tests zeigen, dass die Ergebnisabbildung unseres Algorithmus nahezu konform ist. Wir erweitern unseren Algorithmus, um variables Design zu ermöglichen, das auf Regeln basiert, die auf dem konform abgebildeten Koordinatensystem im Strich agieren.

Contents

Acknowledgment	iii
Abstract	iv
Zusammenfassung	v
1 Introduction	1
1.1 Terminology, research question and contributions	2
1.2 Related work on (decorative) pen simulation	6
1.3 Related work on conformal mappings	13
1.4 Overview of the chapters	17
2 Continuous Model	19
2.1 Definition of a stroke	20
2.2 Envelopes of a family of curves	26
2.3 Limiting envelope of a family of circles	29
2.3.1 E_ℓ for constant radius	32
2.3.2 E_ℓ for non-constant radius	33
2.4 Discriminant envelope of a family of circles	35
2.4.1 Equality of discriminant and limiting envelope	37
2.4.2 Projection of a manifold	38
2.4.3 Local structure of E_d	40
2.5 Criteria for singularities of the envelopes	43
2.6 Type of points of regression in the envelopes	45
2.6.1 A_2 singularities of discriminant envelopes	46
2.6.2 $A_{\geq 3}$ singularities and a connection to catastrophe theory	54
3 Conformality	62
3.1 Conformality of Jordan domains	63
3.2 Quadrilaterals and conformal modulus	64
3.2.1 Properties and computation of the conformal modulus	69
3.2.2 Conformal map of an upper half annulus to a rectangle	72
3.3 Extension of conformal maps to the boundary and beyond	74
4 Discrete Data and Conformality	77
4.1 Interpolation of curve γ and radius function r	78
4.1.1 Linear interpolation and cubic Hermite splines	78
4.1.2 Cubic B-splines	80
4.1.3 Discrete equidistant points on the interpolated curve γ	84

4.2	Discrete conformality	86
4.2.1	Discrete conformal equivalence of polyhedral surfaces	88
4.2.2	Circle packing	91
4.3	Non-univalent discrete strokes	94
4.3.1	Detection of self-intersection	94
4.3.2	Identification of singular envelope points	96
5	Implementation: Pixel Averaging on the GPU	101
5.1	General algorithm	103
5.1.1	Initial coordinates for pixel averaging	104
5.1.2	Storing coordinate information on textures	107
5.1.3	Reverse pixel lookup	110
5.2	Pixel averaging	111
5.2.1	Pixels inside the stroke	112
5.2.2	Boundary handling	114
5.3	Test for conformality and stopping criterion	119
5.3.1	Pixel based test for conformality	119
5.3.2	Local minimum of average error	123
5.3.3	Comparison of conformal moduli	127
5.4	Adaptions for non-univalent strokes	128
5.4.1	Extension of the algorithm for self-intersections	128
5.4.2	Extension ideas for singular boundary points	131
6	Application: Ornamental Design	135
6.1	Tile recognition in the algorithm	137
6.2	Use cases of variable tiles	140
6.2.1	Floral design: radius dependent textures	140
6.2.2	Braid design: transitions and variations	143
6.2.3	Further design concepts	146
7	Future Research	148
	List of Figures	151
	Bibliography	154
A	Conformal Map of an Ellipse to an Infinite Strip	161
A.1	Jacobi elliptic sine function: map from an ellipse to the unit disk	163
A.2	Concrete calculations	168
B	Logarithmic Spirals	171
B.1	Tangential logarithmic spirals	172
B.2	A CPU based spiral pen	178

1 | Introduction

Ornaments have been used for decoration throughout human history. Since the digital age, there are several applications that help to create ornamental designs digitally. In this thesis, we take a digitally created ornamental design as input, implement a stroke model for an interactive digital calligraphic pen, and present an algorithm that computes a nearly conformal map of the ornamental design to the stroke while it is drawn. This requires a combination of different mathematical topics and the interplay of different disciplines, as indicated below.

Figure 1.1 shows a visualization of the basic concept behind our algorithm. The input domain consists of repeated ornamental units that are then mapped to a stroke. The stroke model is based on a user’s drawing on a digital drawing surface, i.e., on the path that is drawn with a cursor on a computer or with a digital pen or finger on a tablet. The width of the stroke varies depending on the amount of pressure applied to the pressure-sensitive pen. Our algorithm maps the ornamental design to the stroke in a mathematically sound and aesthetically pleasing way while the stroke is drawn.



Figure 1.1: Mapping an ornamental strip to a simulated pen stroke.

One of the main challenges is the uncertainty about how the stroke will continue. Therefore, a complete study of the stroke model from the mathematical side is necessary to understand its structure and the cases that may occur. To comprehensively analyze the stroke model, we use results from differential geometry, singularity theory, and catastrophe theory.

Another challenge is to make the computations local and very fast, so that the algorithm can compute the map to the ever-growing stroke simultaneously to the drawing process. To meet the constraints of real-time and locality, our algorithm is written for the graphics processing unit (GPU). Additionally, we make the computational process, i.e., the adjustments to the mapped ornamental design, visible by using the graphical textures on which the GPU performs the calculations.

Furthermore, the algorithm maps the ornamental content of the rectangular input domain to the stroke in such a way that the map is structurally correct, i.e., that the

underlying mathematical properties of the model as well as conformality of the output are met. To achieve this, we study the theory of discrete and continuous conformal maps.

The question of how to compute conformal maps has been studied since the early nineteenth century. Their history is formed by many mathematicians who developed different approaches and answers to the question how to compute conformal maps between different types of (non-)planar domains. Since this turns out to be a difficult task, researchers started to approximate conformal maps and found discrete counterparts to the existing continuous theory. Although conformality is a well-studied area, new results on conformal maps and algorithms for their (approximate) computation are still being explored, as it is the case in this thesis.

In the following, we first formalize the research question and state the contributions of this thesis. Then we give an overview of existing results in related areas of digital pen simulation and conformal mappings. The introduction concludes with an overview of the contents of the individual chapters of this thesis structured as a “how to read” for readers interested in specific aspects of our work.

1.1 Terminology, research question and contributions

The ornamental input to our algorithm consists of copies of a rectangular image T . Multiple copies of T aligned in one direction in the plane define a rectangular strip. If an infinite number of copies of T are aligned, we get an infinite strip S . A finite strip S_n for $n > 0$ contains $[n]$ full copies of T and possibly a truncated last copy if $n \notin \mathbb{N}$. Thus, the width of a finite strip S_n depends on the dimensions of T and the real number $n > 0$. Figure 1.2 shows a strip S_n with $n = 4.5$.

We call the rectangular image T a *tile*, and since it usually contains some artistic motif, we also call it an *ornamental tile*. In general, the tile T can contain any artistic content. However, if the image is a generating tile of a frieze pattern, i.e., it has translational symmetry in the directions the copies of T are placed side by side, as in Figure 1.2, the resulting design within the strip may look more pleasing.

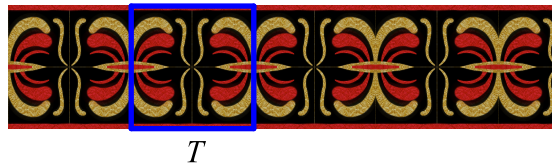


Figure 1.2: Finite strip consisting of $n = 4.5$ ornamental tiles T .

Definition 1.1.1 ((Infinite) strip)

Let T be a rectangular tile of height h_T and width w_T . Then the finite strip S_n consisting of $n > 0$ copies of T is defined as

$$S_n := \{(x, y) \in \mathbb{R}^2 \mid 0 \leq x \leq n \cdot w_T, 0 \leq y \leq h_T\}.$$

If an infinite number of copies of T are placed side by side, the result is an infinite strip

$$S := \{(x, y) \in \mathbb{R}^2 \mid 0 \leq y \leq h_T\}.$$

The artistic contents of the rectangular strip are mapped to a calligraphic stroke that is modeled in a way that adequately supports the conformal map. Let the curve $\gamma: I \rightarrow \mathbb{R}^2$ for a closed interval I be drawn on a tablet using a pressure-sensitive pen. Then the pressure applied is converted to a radius function $r: I \rightarrow \mathbb{R}_{>0}$. This produces a family of circles with centers at $\gamma(t)$ and radii $r(t)$ for all $t \in I$. The stroke is defined to be the union of all these circles. Figure 1.3 on the left shows a stroke defined by circles along the blue curve γ . Our stroke model is not intended to provide a realistic feeling brush stroke. Instead, the objective is to provide a stroke that is fully mathematically modeled in order to define a mapping algorithm that is adapted to this model.

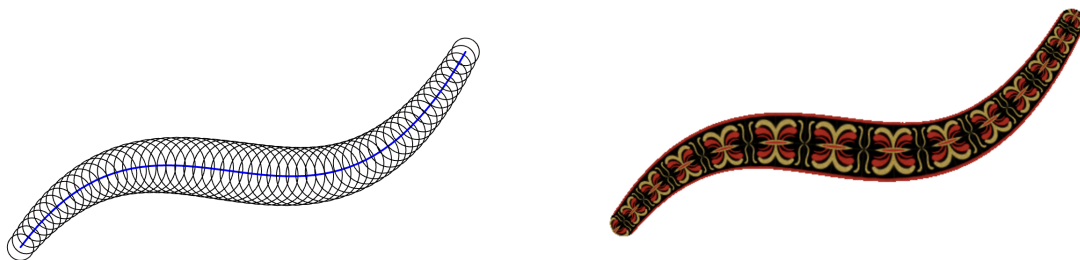


Figure 1.3: A family of circles along a curve γ and the result of our algorithm.

Since our algorithm depends heavily on the model of the digital pen stroke based on families of circles, we study families of circles with techniques from differential geometry and derive insightful results about their characteristics. In particular, we study the properties of the envelopes of families of circles, which essentially are the boundary of a stroke. Although envelopes of families of circles can be defined in three different ways, our detailed analysis shows that they are essentially all the same in our use case. We show that the boundary of a stroke for a regular curve γ is regular, except for sparsely located singular points. Concrete criteria on the curvature of γ and the radius function r are developed, which determine the position of singularities on the envelope of a family of circles. We prove that these singular points are typically cusps, although in certain situations they can be points of infinite curvature of the boundary. This follows from a connection between envelopes and swallowtail surfaces from catastrophe theory.

The algorithm is supposed to calculate a conformal map of an artistic strip region to a user-drawn stroke as above in real-time. Figure 1.3 on the right shows a result of our algorithm. The rectangular ornamental strip has to be mapped to the stroke without apparently deforming the artistic content or obviously piecing together several parts of the strip. The goal of our map is to appear organic and preserve the artistic content of the strip so that it is locally distorted as little as possible, while perfectly fitting the simulated stroke along the user-drawn curve γ .

There are several metrics used to measure image distortion, including area preservation and angle preservation, commonly known as conformality. Which metric property is best for achieving the desired result always depends on the individual task or topic. A classic example in this context is the mapping of the Earth to a flat map. Preservation of angles is important for ship navigation, while area preservation is essential for land surveys. However, it is impossible to combine both properties in one map. In this thesis, we

choose angle preservation as the decisive metric and strive for a map that is as conformal as possible.

In accordance with Penrose [Pen06, p.139], Needham [Nee23, p.193 ff.] and Stephenson [Ste05, 3.4], a conformal map is defined to be angle and infinitesimal shape preserving.

Definition 1.1.2 (Conformal map)

A conformal map is a map between two regions in the plane or between two surfaces in space that preserves oriented angles between curves and infinitesimal shapes. Hence, locally a conformal map is a similarity transformation.

Penrose gives a good visual description of this definition: “We can think of this applying to small (infinitesimal) circles drawn on the plane. In a conformal map, these little circles can be expanded or contracted, but they are not distorted into little ellipses.” [Pen06, p.139]

Since conformal maps preserve angles, our algorithm generates a conformal image of a strip in the stroke that deforms the ornament by only a small amount according to this angle measure. Additionally, cutting and gluing together ornamental tiles would contradict the preservation of angles and infinitesimal shapes. Furthermore, conformal maps are explicitly defined between the surfaces or regions involved which ensures that the artistic content of the strip is mapped inside the stroke.

Calculating a concrete conformal map between two domains can be a challenging task. The famous Riemann mapping theorem [LV73, 2.1] states that a conformal map between two simply connected regions of the plane always exists, which is a very powerful statement. However, the proof is not constructive, i.e., it does not state how to find such a map. There is extensive completed and ongoing research in the area of conformal maps, which we partially review in Section 1.3. The majority of approaches find numerical solutions to parameter problems, compute minima of energy functions, or examine very specific domains. Despite some of the numerical methods are rather fast, they are not applicable to our task since they are not constructed to work on constantly changing regions as required by our algorithm. Our algorithm is supposed to run on user-drawn strokes in real time, while showing the ornament flow into its intended conformal position. Moreover, we cannot expect to get a target domain that is in an analytically simple form, since the stroke is drawn by a user and does not follow a known equation. Nevertheless, we can generate a conformal map based on almost any user input, including self-intersecting strokes that are not simply connected.

We require the algorithm to respond immediately to the user input. Consequently, when the user draws a curve on the screen or tablet, the mapped ornamental design is progressively shown during the simulation of the stroke. This raises the question of how long the strip S_n must be to be conformally mapped to the stroke. This cannot be known before the drawing is completed, because the final appearance of the stroke is unknown until the user stops drawing. Therefore, our algorithm computes the conformal map of the strip onto the stroke adaptive to the stroke’s appearance and acts locally to make calculations parallel to the drawing process possible.

To ensure fast calculations that can keep up with the drawing of the stroke, the algorithm is specifically designed for the graphics processing unit (GPU) of modern computers.

GPUs are built to work massively in parallel, which means that calculations are both fast and local. A GPU algorithm operates on the pixels of a graphical texture, which is why we store the stroke domain and rectangular ornamental tiles in graphical textures.

The algorithm computes the preimage coordinates in the ornamental strip for each pixel within the stroke using a method known as *pixel averaging*. For initialization, each pixel in the stroke is assigned a reasonable preimage coordinate within the strip. These coordinates are subsequently averaged by calculating the arithmetic mean of the preimage coordinates of neighbors for all pixels in the stroke. Repeatedly applying this averaging step causes the map to converge, which leads to a conformally mapped coordinate system that is transferred from the strip to the stroke. Figure 1.4 provides a visualization of such a mapped coordinate system.

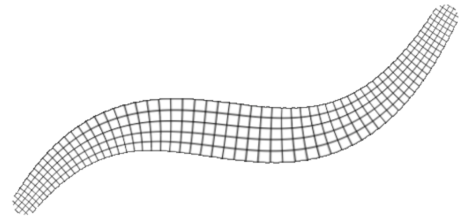


Figure 1.4: A coordinate system within the stroke.

Such a pixel averaging algorithm has previously been presented by Swart [Swa11] where maps are computed on the central processing unit (CPU) between predefined planar regions. Koczyński et al. [KCK19] used a similar algorithm to Swart [Swa11] to find mappings from the hyperbolic plane to simply connected domains with long and narrow regions. Montag and Richter-Gebert [MRG20] have developed a pixel averaging algorithm that works on the GPU for planar regions bounded by line segments and arcs. To extend the map along the boundary, they use the Schwarz reflection principle. Their algorithm is applied, for example, to find seamless maps from ornamental tiles to a tiling of the hyperbolic plane. Reinhardt has demonstrated in his thesis [Rei23] that the speed and quality of the pixel averaging algorithm are influenced by the resolution of the region’s underlying texture, as well as the number and selection of neighboring pixels for each pixel.

The main contributions of this thesis to the pixel averaging algorithm are the following: we have adapted the pixel averaging procedure to cope with constantly changing, i.e., growing, regions as we have suggested in [Pol23a]. The algorithmic calculations are applied to the strokes as they are drawn, so that the target region of the desired conformal map is constantly changing. To obtain the desired artistic design in the stroke, it is necessary to adapt the width of the preimage strip to the changing stroke. Moreover, the user-drawn strokes are not only bounded by circular arcs or line segments. Therefore, we have extended the pixel averaging algorithm using a generalization of the Schwarz reflection principle for conformal maps. This generalization allows to compute a nearly conformal map of the coordinate system and of the design in a tiled strip to the stroke. In particular, we have modified the pixel averaging algorithm to ensure it can handle self-intersecting strokes although they cover several parts of the drawing surface more than once.

To evaluate the resulting map from the tiled strip to the stroke in terms of conformality, we have developed a visual and numeric test system that works in real time simultaneously to all computations. For this, we have adapted the method used by Reinhardt [Rei23] and test for local angle and infinitesimal shape preservation by comparing the angles and length-cross-ratios of infinitesimal quadrilaterals in the preimage and target domain. The

results for each pixel are visually indicated by a gradually changing color scheme that overlays the mapped image. As an illustration, the green color shown on all pixels of the stroke in the right picture of Figure 1.5 validates that our algorithm has successfully approximated conformality for the stroke depicted on the left without superimposed color.

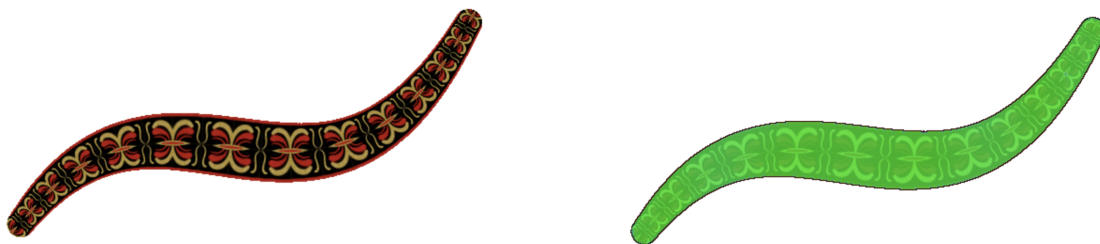


Figure 1.5: Testing the outcome of our algorithm for conformality.

In addition to the local and visual test, we have devised a method for computing the mean deviation from exact angle and shape preservation for the entire stroke. This method also serves as a stopping criterion, indicating when the pixel averaging algorithm has reached a local minimum of conformal deviation and can be stopped.

Another achievement of our work is the adaptation of our own algorithm to compute conformal maps beyond using tiled ornamental strips as preimages. We have developed a framework that allows to establish rules which define adaptive design for the stroke based solely on the computed conformal coordinate system. These rules can be based on features such as the pressure sensitivity of the stroke simulation or the variable length of the drawn stroke.

1.2 Related work on (decorative) pen simulation

In this section, we provide an overview of the existing literature regarding the simulation of pens and brushes. Some of them may appear to satisfy the requirements on our stroke model and the associated conformal map of an ornamental design to the stroke. However, the subsequent analysis shows that the presented digital pens are not able to adequately solve our task of conformally mapping the given ornament in real time to the constantly changing user-drawn stroke. This collection is by no means complete. Further related research may be found in the references of the cited sources.

Brush simulation

There is a vast field of research concerning the simulation of realistic brush pens. This research includes the behavior of the brush hair, the distribution of ink or other paint, and the artist's brush guidance. The goal of our pen simulation is not to model a realistic physical pen. We are aiming for hyper-realistic pens whose behavior is generally digital, but which still produce intuitively reasonable output. We give an overview of some

approaches to simulate realistic physical pens, although we actively decided against this approach for our pen model. Nevertheless, it might be interesting in the future to combine oriental brush modeling with our method of real-time conformal mappings.

A virtual Chinese calligraphy brush model has been developed by Wong and Ip [WI00]. It takes into account properties of real calligraphy styles, such as brush hair properties and ink behavior on the path depending on, for example, drawing speed and amount of ink on the brush. The physical 3D movement of the brush tip and stem is simulated, and ellipses are drawn at the intersection of the brush and paper. The user can first draw the stroke paths and then specify the pressure and orientation of the drawing. It is also possible to change the stroke after setting all the parameters of the simulation by stretching or rotating the stroke, creating even more variety in calligraphy style. The model does not operate in real time.

A real-time model was introduced by Xu et al. [XLTP03]. They developed a virtual brush for interactive digital painting with the support of offline components. The system is based on the physical behavior of brush hair, which covers complex effects of real brushes such as brush splitting. The behavior of the paint pigments is modeled with a diffusion process of random molecules, using lookup tables to solve the diffusion equations as an offline component. A unique feature of the model is a component that adapts to the user’s painting habits. This makes the system attractive for beginners, since the user does not need to practice much before being able to use the simulation accurately to paint to his or her satisfaction. This is achieved by allowing the user to select from a set of stroke patterns that are used to train the offline model. The real-time simulation then generates output strokes that are similar to those that the user has put into the training.

Another real-time model was presented by Guo et al. [GHY+15]. Here, the relationship between applied force and brush deformation is analyzed, where the force information is sent back to the user to simulate the feeling of drawing with a Chinese brush via a “phantom desktop haptic device”. A spring-mass model is used to model the behavior. For this purpose, a spring is inserted into the model perpendicular to the drawing plane at the transition between the brush handle and the hair. The deformation of the spring along this normal is used to calculate the force and imprint of the brush on the paper in real time. The imprints depend on the pressure applied, the simulation of flattening, and the movement of the bristles when force is applied to the brush. These imprints are superimposed to obtain the brush stroke along the sample points.

Gong et al. presented a Chinese brush simulation [GNH+17] that is not based on a physical model, but on experience with brush models to achieve less computation time. The model includes real-time stroke generation and ink dispersion. In addition to the real-time property, the model has a second commonality with our stroke model: circles are placed along the drawn path to get a stroke model. At low drawing speeds, a Kalman filter algorithm handles non-smooth points; at high drawing speed, interpolation between points is performed using arc and linear interpolation. A modified Lattice Boltzmann equation algorithm, based on fluid mechanics and computed on the GPU, is used for the ink dispersion of the initial brush stroke.

A model that differs from the others in its approach was presented by Wu et al., where a neural network is used for an oil painting simulation [WCW+18]. The goal is to reproduce the details found in real painting with real brushes that artists are usually able

to achieve when working with non-digital material. The model achieves results comparable to accurate physical oil painting simulations, but it is faster and less expensive to compute. To achieve these results, a generative adversarial network (GAN) is trained on a dataset obtained from a high-quality oil painting simulation with an accurate fluid solver, many brush bristles, and high canvas resolution. Images of real oil painting strokes are not used as input because of the lack of sufficient real training data due to the difficulty of capturing images of pressure and bristle movement. To add a new stroke, the GAN takes as input the painted canvas with height map and color information, and the path of the new stroke with its bristle pressure trajectory and color information. It returns the height and color information of the new stroke. After training the model offline, the GAN works fast enough to provide a real-time oil painting tool.

The last real-time brush model we want to mention is the Japanese calligraphy brush model for ink painting by Shin et al. [SKR19]. Its main feature is the ability to model the two different types of pressure that an oriental brush has: the pressure of the brush tip and the pressure of the pen stem, which is different because the brush hairs of this type of pen are very flexible and soft. The pressure of the brush base on the paper is measured by the xy -coordinate, pressure, direction and altitude registered by the digital drawing device. The pressure on the tip of the pen is measured by the z -coordinate, which is recorded by a webcam. There is a droplet model for the brush simulation which places ink droplets for each drawn point. The color and appearance of the droplets depend on the different pressures: light ink is applied to the entire brush, and shading ink is applied to the tip of the pen, creating interesting color effects depending on the proportion of the brush tip and stem contributing to the drawn droplet according to the measured pressure. Scratchiness, i.e., the loss of ink when drawing a stroke, and ink diffusion are influenced by the amount of water and ink density, which are also captured by the model.

Decorative pens

Besides models that simulate the behavior of real brushes, taking into account the deformation of the tip of the pen, the movement of the bristles and the distribution of ink on the paper, there are also programs that create decorative pens. There are many forms of decorative pens, such as the so-called *skeletal strokes*, the *deco brush* and the *pattern brush*. Again, more related work on this topic can be found, for example, in the citations of the discussed literature.

So-called skeletal strokes were introduced by Hsu, Lee and Wiseman [HLW93]. Their model, like ours, is based on two main objects: a user-drawn path and a reference image. In their model, however, the reference image has a reference line, the *spine* or *backbone* of the image, which is mapped to the user-drawn path. The image around the spine is supposed to be mapped around the user-drawn path “by bending, shearing, twisting, while preserving the aspect ratio of selected features” [HLW93, Abstract] of the reference image. This deformation of the image is not conformal and thus different from our approach. There are two stroke styles to choose from: one keeps the shear angle of the image around the path constant, while the other keeps the shear angle with respect to the path, which makes the result look like it was drawn with a flat nibbed pen. The user can decide where certain parts of the image are mapped by anchoring them. To stretch parts of the image

by different amounts, the user can also set reference points on the backbone and the path, respectively. This is another difference to our approach, since the user first draws a path and selects the aforementioned anchor points or the like before the reference image is mapped to the stroke. Thus, the image is not mapped in real time while the user draws the path.

Hsu and Lee use this skeletal stroke model to generate cartoon drawings and animation [HL94]. Different brush stroke images are used as reference images, creating the visual impression that different pen types were used within a single artwork. The backbone of the reference image and the selected stroke thickness provide a parameterization of the coordinate system within the reference image. This coordinate system is deformed to match the stroke, which corresponds to our goal of mapping the coordinate system of the tiled strip to the stroke. However, the skeletal stroke method intentionally does not operate in real-time since the authors “are usually more interested in the final appearance of the stroke than the physical action of dragging a brush across paper” [HL94, 3.3].

Several years later, Asente studied the behavior of skeletal strokes in folding regions, as presented in [Ase10]. Again, a reference image with a spine is specified and mapped to a user-drawn path. The user defines the width of the target stroke, and so called *ribs* are placed along the reference spine and the drawn path. These ribs are perpendicular line segments that tile the reference image into rectangular slices. The ribs are mapped to line segments that intersect the target path perpendicularly. Problems occur when the radius of curvature of the target path is smaller than half the width of the ribs, i.e., half the width of the stroke, or where the path has a corner. In these cases, neighboring ribs intersect and the mapped image folds onto itself. We will address similar problems in Chapters 2, 4 and 5. In his paper, Asente relocates ribs within the problematic area of the stroke (see Figure 1.6a) so that they all pass through the center of crossing, which is where the stroke envelope intersects itself. As a result, the ribs in the fold region may be elongated and no longer perpendicular to the target path. See Figures 1.6b and 1.6c for visualizations of relocated ribs. However, the reference image is still mapped to the stroke such that the spine is mapped to the path and the ribs are mapped to the potentially adjusted ribs. As a result, the image becomes even more distorted and it is surely not conformal.

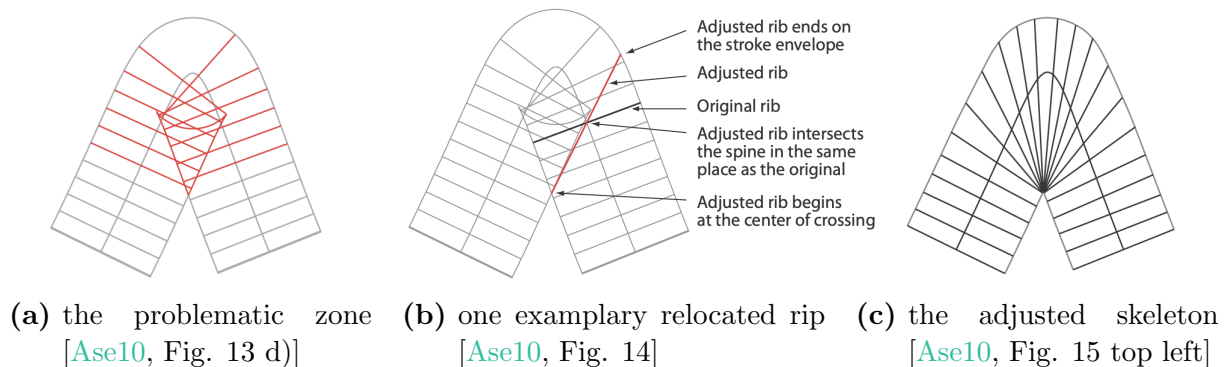


Figure 1.6: Adjustment of folding regions in skeletal strokes [Ase10].

Another difficulty mentioned by Asente is that the setting of a spine and sparsely distributed ribs does not uniquely define the stroke’s envelope when the path makes a corner. In Figure 1.6, the ribs on the top of the corner are connected by a rounded curve, but the envelope could also be beveled by a line segment connecting the ends of the ribs, or it could make an angular corner itself. However, the method works with any convex solution for this situation. In contrast to that, there is no need to guess the envelopes in our model, since the stroke model of a family of circles naturally has a unique envelope, which we study in detail in Chapter 2.

Another model, which also treats curves equipped with a ribbon of constant width around them, is presented by Zhou et al. [ZLL13]. The ribbon is sliced along its medial axis, and potentially tilted pieces from a sample pattern are placed on the ribbon slices. The content of the sample pattern is not distorted. On the contrary, the synthesizer copies these pieces of the sample pattern and creates a new pattern specifically for the given curve, which is visually continuous. To achieve this continuity, a cost function compares the neighboring pixels of a piece’s right edge to its neighbor’s left edge and chooses an optimal deformation at the edge. The deformation is then propagated into the interior of the pattern pieces, and finally the contents are interpolated along the edges. Figure 1.7 shows some results. Except for the use of the GPU and the result of a decorated stroke, this approach has little in common with our goal, since it alters the sample pattern by cutting and gluing it together. Our model preserves the artistic content of the ornamental design by deforming it as little as possible. However, the model of Zhou et al. is able to handle closed curves, which our approach cannot handle so far. This would be an interesting task for future research.



Figure 1.7: Exemplary results of synthesized pattern from [ZLL13, Figure 7].

Another non-real-time model called *DecoBrush* was presented by Lu et al. [LBW⁺14]. In this model, the user selects a sample collection of decorated strokes from a library and then sketches curves that indicate the approximate layout of the resulting strokes. Once

the sketches are complete, they are subdivided into segments that correspond to similar segments among the sample strokes. The sample strokes are equipped with a start and end point and a curve connecting them, like the spines of skeletal strokes. The given points and the curve are now adjusted to the segments of the user-drawn path to attain that the deformation is as rigid as possible. This is accomplished by modifying the sketched path to better fit the stroke samples, which is different from our approach where, the user-drawn path remains as unmodified as possible in order to create the stroke model. When different sample strokes are placed on one sketched path, the decorated pattern may overlap at the junctions or fail to fit seamlessly. To address this issue, graph-cuts and texture synthesis are used, and users have the option to modify the result if it is not to their satisfaction. This also differs from our approach, where the ornamental strip is seamless. The result of our algorithm is determined by the drawing process and cannot be changed afterwards. Another aspect of *DecoBrush* is that it is able to merge the pattern structure in the case of a self-intersecting sketched curve. This is not yet possible in our approach. However, it would be highly desirable to address this aspect in future research.

The software *Adobe Illustrator* provides a collection of different brushes. The user manual [Inc19, p.270 ff.] mentions a *calligraphic brush* that creates a monochrome stroke resembling the stroke drawn with an angled tip of a calligraphic pen. The brush’s angle of rotation, roundness, and diameter can be altered, possibly in response to the pressure, tilt, bearing, and rotation of the pen device under the assumption that it can detect such data. The other pens have comparable settings that may be modified by the same properties of the digital pen. A *bristle brush* aims at the simulation of a realistic brush stroke with real paint, e.g., watercolour. The user’s drawing pressure, tilt, and other factors affect the length, rigidity, and density of the bristles, as well as the thickness and opacity of the paint. This relates back to the brushes presented in the previous section. The *art brush* stretches an artistic sample along the user-drawn path, resembling skeletal strokes. The *scatter brush* places copies of selected objects side by side along the user-drawn path. The following aspects can be modified: the size of the objects, the distance from each other, how closely they follow the path, the angle of rotation of the individual objects and whether the rotation is with respect to the path or the page. This pen does not meet our goal as the objects are simply copied and pasted along the path without being adjusted to follow the path’s course.

The *pattern brush* in *Adobe Illustrator* is of particular interest as it may appear to be similar to our model. However, this is not the case. Adobe’s pattern brush utilizes tiles as input, which are then repetitively applied along the path of the user’s drawn curve. The curve is initially outlined as a dotted path, and the selected tiles are then adjusted accordingly. Subsequently, control points can be dragged into place to modify the stroke. The user may choose from a selection of up to five tiles, with one tile assigned to the start and end of the stroke, one for an inner corner, one for an outer corner, and one for the regular stroke segment. The mapping of these tiles to the corresponding segments of the stroke involves simple stretching and shortening of the tile. Figure 1.8 shows several strokes drawn in *Adobe Illustrator*, which demonstrate how different the *pattern brush* is from the result of our algorithm. In the top left, a special corner tile is used that becomes deformed when the corner is sharp. In both examples on the right side, a pattern tile is shortened. On the top right, we can see a massive deformation of the tile on the inner side of the narrow turn of the stroke (notice the small turquoise shape). In the bottom

left example, there is inconsistent use of a corner tile and two deformed regular tiles for a stroke that makes an approximately right angle. The example in the bottom right illustrates a loop with a stitching pattern clearly distorted differently on the outer and inner sides of the loop.



Figure 1.8: Pattern strokes drawn with Adobe Illustrator’s pattern brush.

Mech and Miller created another *Deco* framework [MM12], which Adobe integrated into some versions of Flash Pro and Photoshop. The framework uses procedural modelling with predefined rules to control the growth process of decorative structures from underlying *scriptals* within a user-defined area. A *scriptal* is a fundamental module of the structure which contains rules for its own growth and how it interacts with its environment. The *scriptals* develop tree-like structures which contain non-colliding objects. Figure 1.9 illustrates that the user may specify an approximate shape for the model to fill with automatically growing structure (picture on the left). Alternatively, the user can initiate growth by brushing the stems directly (middle example). This produces a decorative brush stroke, filled with self-growing ornamentation (right example). In all cases, however, it is hard to predict the precise contour of the resulting area since it depends on the procedural growth and the chosen parameters.

The *Deco* framework is founded upon the work of Wong et al. [WZS98]. Wong presents rules for creating ornamental pattern, which do not rely on Lindenmayer systems [PH13], even though they may appear to be a natural choice for expressing growth rules. The rules are specified as procedures with parental and child elements, and the system looks for the largest empty space within the designated area and fills it with ornamental design.

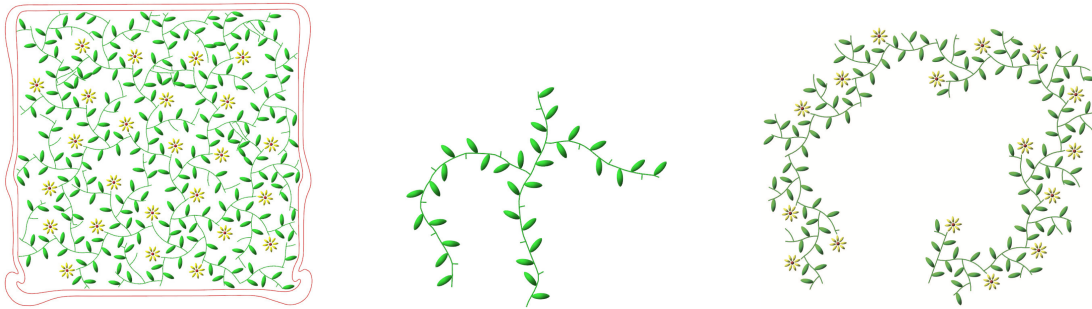


Figure 1.9: Decorative procedural growth from [MM12, Fig. 7].

1.3 Related work on conformal mappings

The history of conformal maps goes back to Gauss in the early nineteenth century [DT02, p.4]. Since then, many mathematicians investigated between which regions conformal maps exist and how they can be computed. Riemann stated in his famous mapping theorem that conformal maps exist between any two simply connected proper subsets of the complex plane. His result is powerful, but not constructive. This section provides an overview of various methods for approximating conformal maps. Additional literature can be found, for example, in the references of the cited sources. All presented methods have in common that they compute conformal maps for given source and target domains that remain constant, unlike our strokes which are constantly changing. The majority of these techniques are incapable of computing conformal approximations rapidly enough for real-time computations. While some algorithms are fast, they do not satisfy the characteristic feature of our approach, namely that the computational process and the convergence to a conformal map are made visible.

One of the best-known techniques is the approach of Schwarz and Christoffel, the so-called *Schwarz-Christoffel mappings* (SCmaps for short). Their book [DT02] provides a dense introduction and overview of the potential of SCmaps, along with Matlab-packages and libraries for calculating conformal maps between planar domains. The core concept behind SCmaps is that the derivative of a conformal map f between two planar domains is the product of other functions: $f' = \prod f_k$. Therefore, integration of the product results in the conformal map f . Traditionally, an SCmap is computed between the upper half plane H^+ and the interior of a simply connected polygon P . The vertices w_1, \dots, w_n of the polygon are ordered counterclockwise on the polygon's boundary and their interior angles are denoted by $\alpha_1\pi, \dots, \alpha_n\pi$ for $\alpha_k \in (0, 2)$ with $k \in \{1, \dots, n\}$. For a closed polygon, the sum of all exterior angles must be equal to 2π , which results in a constraint on the parameters α_k : $\sum_{k=1}^n (1 - \alpha_k) = 2$. The preimages of the vertices w_k of P on the real axis of the upper half plane H^+ are denoted by $z_k = f^{-1}(w_k)$. The conformal SCmap f is then defined by

$$f(z) = A + C \int^z \prod_{k=1}^{n-1} (\xi - z_k)^{\alpha_k - 1} d\xi \quad (1.1)$$

for complex constants A and C , and for $f(\infty) = w_n$ where ∞ is the point at infinity on the real axis. This formula can be adapted if the preimage domain is, for instance, the unit disk rather than the upper half plane or if f maps to the exterior of P , to a circular polygon, to a slit domain, or a doubly connected domain. The primary challenge lies in calculating the unknown preimage vertices z_k , since the choice of the preimage vertices influences the side lengths of the resulting polygon. Nevertheless, any map (1.1) with predefined interior angles $\alpha_k\pi$ maps the upper half plane to some polygon with those interior angles. Therefore, in order to get the intended conformal map f , a parameter problem needs to be solved to find the appropriate preimage vertices z_k for a given w_k . After identifying the preimage vertices, the integral can be calculated for every z inside the domain using Gauss-Jacobi quadrature to obtain the conformal map of the upper half plane to the desired polygonal domain.

There are various methods for solving the parameter problem for preimage vertices z_k . Many of them run into the problem of crowding, which means that the preimage points z_k are so close to each other on the real axis or the unit circle that they can't be distinguished numerically. This problem occurs for elongated regions, for example. Driscoll and Vavasis introduce an algorithm that solves the parameter problem for a conformal map of the unit disc to a polygon with n vertices and overcomes the crowding problem [DV98]. The algorithm is called CRDT which is an abbreviation for “cross-ratio of Delaunay triangulation”. To determine real parameters that define the positions of the n preimage vertices z_k on the unit circle, a Delaunay triangulation of the target polygon is computed and possibly additional vertices are introduced on the polygon edges so that the resulting Delaunay triangulation does not contain long and narrow triangles that would cause crowding problems at the preimage vertices. The Delaunay triangulation of a polygon with N vertices ($N \geq n$ after splitting edges) has $N - 3$ diagonals, i.e., edges that are not edges of the original polygon. Along each of these $N - 3$ diagonals d , neighboring triangles form a quadrilateral $Q(d)$ with four vertices. The CRDT algorithm is based on the fact that Möbius transformations are defined by three pairs of points, and that cross-ratios of points on a circle are invariant under Möbius transformations. The preimage vertices of all N vertices in the polygon are located on the unit circle, and their position is unique up to a Möbius transformation. Hence, three preimage vertices can be chosen freely, and all remaining $N - 3$ preimage vertices are determined uniquely if values of the corresponding cross-ratios are given. The values of the cross-ratios are chosen as the logarithms of the absolute values of the cross-ratios of the quadrilaterals $Q(d)$. With these values, the preimage vertices z_k can be determined without crowding problems since for each of the $N - 3$ values, the three preimage vertices used to determine a yet unknown preimage vertex can be placed on the unit circle in a way that ensures that they are well distinguishable. Using the preimage vertices as input, Driscoll and Vavasis calculate an SCmap from the unit circle to a polygon. They then minimize the difference between the logarithms of the absolute values of the cross-ratios of the quadrilaterals $Q(d)$ in the target polygon and the corresponding quadrilaterals in the result of the SCmap. The result is a map that conformally maps the unit circle to the given target polygon.

Other methods for calculating conformal maps f utilize the characteristic that they can be separated into two harmonic functions u and v : $f = u + i \cdot v$. A real-valued function u is harmonic if $\Delta u := \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$. Harmonic functions are smooth and easy

to compute compared to conformal maps.

Chen and Weber use harmonic functions to compute a map between two planar domains with bounded conformal distortion [CW15]. The user can select a finite number of points on the boundaries of or within the planar regions of the preimage and target domains. Additionally, the user can choose three non-negative real constants that limit the values of conformal distortion and the minimal and maximal local stretch. Conformal distortion is measured by the ratio $\frac{|f_{\bar{z}}|}{|f_z|}$. The map f is conformal if and only if the conformal distortion is zero. To find an optimal bounded distortion harmonic mapping for the given input, a numerical optimization problem is solved. The map f is then given as $f = u + i \cdot v$ for u and v being harmonic functions satisfying the Cauchy-Riemann equations $u_x = v_y$ and $u_y = -v_x$.

Another method involving harmonic functions for simply connected domains with smooth boundaries is presented by Trefethen [Tre20]. For the simply connected domain W bounded by a Jordan curve P , there exists a unique conformal map f from W to the unit disk D such that $f(0) = 0$ and $f'(0) > 0$. The map f can be expressed as $f(z) = z \cdot e^{(u(z)+i \cdot v(z))}$, where u is the unique harmonic solution of the so-called Dirichlet problem, i.e., of $\Delta u = 0$ with $u(z) = -\log(|z|)$, and v is the unique harmonic conjugate of u . To find approximate solutions for the Dirichlet problem, Trefethen proposes several methods that are based on the calculation of coefficients of analytic functions. If P is sufficiently well-behaved, u can be approximated by a series of polynomials. However, if P is not close to a circle around the origin, the problem may be ill-conditioned. In such cases, a so-called Arnoldi factorization should be used instead of a least-squares method to find the polynomial coefficients. In addition, the degrees of the polynomials needs to be in the thousands to avoid crowding problems. Another approach is presented for simply connected domains with corners z_1, \dots, z_K . The harmonic function u is approximated by

$$u(z) \approx \operatorname{Re} \left[\sum_{j=1}^{n_1} \frac{a_j}{z - z_j} + \sum_{j=0}^{n_2} b_j \cdot z^j \right]$$

with complex coefficients a_j and b_j . The rational functions in the first sum converge root-exponentially in this case. For the rational approach, the conformal maps are usually computed by rational functions of degree less than 100. However, it is possible that the presented method fails if the sampling of the boundary P is not fine enough or if the samples are not clustered as fine or finer than exponentially clustered poles near the corners on the boundary P .

Hakula et al. compute a conformal map from a simply connected quadrilateral domain Q , which is bounded by four Jordan arcs, to a rectangle using the harmonic function u [HQR13]. The modulus $M(Q) = h$ of Q defines the class of rectangles to which Q can be conformally mapped as those rectangles with side-length-ratio h . In general, computing the conformal modulus of a domain is challenging. However, the unique harmonic solution u to the Dirichlet-Neumann problem on Q can be used to compute h from Q . For this, let z_1, z_2, z_3, z_4 be the four corners of Q . If u equals zero on the arc between z_2 and z_3 , equals one on the opposite arc between z_4 and z_1 , and has vanishing derivative with respect to the normal on the other two Jordan arcs, then h is computed by integration over the boundary of Q and u :

$$h = M(Q) = \int \int_{\partial Q} |\nabla u|^2 dx dy.$$

If \tilde{u} is the solution of the conjugate domain \tilde{Q} with the same vertices but in a shifted order z_2, z_3, z_4, z_1 , then the conformal map f can be expressed as $f = u + i \cdot h \cdot \tilde{u}$.

The conformal modulus is an important element of the domain decomposition method (DDM for short) presented by Falcão et al. [FPS01]. The DDM approximates conformal mappings of elongated quadrilaterals Q with vertices z_1, \dots, z_4 in counterclockwise order to a rectangle that belongs to the same conformal class. The DDM initially decomposes the quadrilateral Q into smaller quadrilaterals Q_j . It approximates the conformal modulus of the entire domain using the conformal moduli of the smaller quadrilaterals. Finally, it determines the approximate conformal map f of Q to a rectangle using the conformal maps of the Q_j to their corresponding rectangles. This method was known to work for domains bounded by two parallel straight lines and two Jordan arcs. However, Falcão et al. extended the DDM to more general quadrilaterals, which include a domain Q that is partitioned by a straight line ℓ into Q_1 and Q_2 such that the reflection of Q_2 along ℓ is a subdomain of Q_1 . In each of the presented cases, the deviation between the approximate conformal map and the exact one is bounded from above.

Delillo approximates a conformal map f from the interior of the unit disk D to the interior of a Jordan domain bounded by a Jordan curve g [Del94]. For $z \in D$, the conformal map is given by

$$f(z) = \sum_{k=0}^{\infty} a_k \cdot z^k$$

where a_k are the Taylor coefficients of f expanded about 0. An approximation of f is provided by a truncated Taylor series F_N of degree N . If the Jordan curve is interpolated by a cubic spline, the approximation error is bounded by $\|f - F_N\|_{\infty} \leq C \cdot N^{-\frac{5}{2}}$ for the N -th Taylor series. To obtain a good approximation of f , N should be greater or equal to $\frac{2\pi \cdot \|f'\|_{\infty}}{L}$ where L is the arclength of the Jordan curve g . Hence, the accuracy also depends on the shape of the Jordan domain since the error scales with the length of the domain's boundary. Furthermore, Delillo presents different approaches and options of Jordan domains, analyzing their behavior with respect to the crowding problem.

Marshall and Rohde show the convergence of the so-called “zipper algorithm” [MR07]. This algorithm computes a precise conformal map f from the upper half plane to a Jordan domain and its inverse f^{-1} . To have f and f^{-1} available is advantageous for many applications. The core part of the zipper algorithm consists in mapping two consecutive vertices z_0 and z_1 of the Jordan domain to the real axis such that the image of the Jordan domain is a subset of the upper half plane: $z_0 \mapsto \infty$ and $z_1 \mapsto 0$. Then the image of the next vertex z_2 is considered. Visually speaking, the edge between 0 and z_2 is split in two, the vertex z_2 is mapped to the origin and the edge is unzipped to both sides of 0 on the real axis. This process is then repeated until all vertices are mapped to the real line and the Jordan domain is mapped onto the complete upper half plane. Disk chains on the boundary of the Jordan domain are used to show convergence of the zipper algorithm.

Weber and Gotsman allow discontinuity at specified boundary points of their mapping between planar shapes [WG10]. For this, efficient algorithms are developed using

barycentric coordinates. This is, the coordinates of z in the preimage region are expressed as linear combinations of coordinates of vertices of the preimage region. Then, the coordinates of the image point $f(z)$ under the conformal map are represented as a linear combination of the vertices of the target shape with the identical coefficients. Weber and Gotsman use a generalization, namely Hilbert coordinates, which help to efficiently compute the conformal map. One drawback is that image points may lie outside of the target shape. To address this issue, it is useful to relax the requirement that boundaries must be mapped to the given shapes and introduce specifications on the angle change.

The method outlined in [SBC16] directly maps between source domain S and target domain T , without using the unit disc or the upper half plane as intermediate region. The *alternating minimization algorithm* presented by Segall and Ben-Chen initially computes a conformal map that maps S to a domain P with approximately the same boundary as T . This map is refined until the boundary of P fits the boundary of T . To quantify the refinement, an energy function is defined on the boundaries which measures the distance between the boundary points on P , i.e., the image of the boundary of S under the current map, to the boundary of T . Alternately, the algorithm minimizes the energy with respect to the currently used conformal map and with respect to the correspondence of the boundary points of T and the boundary points of the current image domain P . It is possible to include a point-to-point or curve-to-curve requirement in the algorithm, defining points or curves in the source and target domains that are to be mapped to one another.

1.4 Overview of the chapters

In Chapter 5, we will present our algorithm which calculates a nearly conformal map between an ornamental rectangular strip and a user-drawn stroke. In Sections 5.1 and 5.2, we will cover the algorithmic details for univalent strokes, i.e., strokes without self-intersections or singular boundary points. The underlying continuous stroke model will be examined in Chapter 2. Sections 2.1, 2.2 and 2.3 will provide the necessary foundations for univalent strokes: the definition of a stroke as a family of circles along a curve with an associated radius function, the definition of the boundary of a stroke, and concrete equations for all points on this boundary. However, digital drawings are not continuously registered by a computer but rather as a discrete set of data points. Hence, Section 4.1 will use interpolation to convert the continuous stroke model into a discrete model. The interpolation method examined in Section 4.1.2 and the algorithm presented in Section 4.1.3 will be used in the implementation of our digital pen, while Section 4.1.1 will provide additional information about interpolation methods and why they are not used in our model.

We will investigate the theory of continuous conformal maps in Chapter 3 in order to define our conformal mapping algorithm mathematically correct at the boundary of strokes. For this, Section 3.1 will state uniqueness results for simply connected bounded planar domains and Section 3.3 will investigate important results concerning the extension of conformal maps to the boundary of such domains. This knowledge will be implemented in Section 5.2.2, which will discuss our algorithm along the boundary of univalent user-

drawn stroke domains.

In Section 5.3, we will present our approach to testing our algorithm for conformality. There will be three different levels of testing for conformality: a local test that is visualized during the drawing process (Section 5.3.1), a global measure that serves as stopping criterion for the pixel averaging algorithm (Section 5.3.2), and a comparative evaluation where an exact conformal map is compared to a benchmark stroke (Section 5.3.3). The fundamental principles for all tests will be outlined in Chapter 3 and in Section 4.2. In Section 3.2, we will analyse the properties of the so-called conformal modulus, which partitions conformal mappings into equivalence classes. Section 3.2.2 will present the exact conformal mapping of a half annulus to a rectangle, which will then be compared to a stroke in the shape of a half annulus. Section 4.2.1 will introduce methods from discrete conformality that will be applied to formulate both our local and global measures of conformality.

In Section 5.4, non-univalent strokes will be addressed, covering strokes with self-intersections and singular boundary points. Regarding self-intersections, Section 5.4.1 will extend the mapping algorithm for univalent strokes and the local and global conformality test. The theoretical basis for detecting self-intersecting strokes will be established in Section 4.3.1. The ideas for the concept originate from the discrete conformal theory of circle packings, which will be presented in Section 4.2.2.

Strokes with singular boundary points will be examined in detail in Sections 2.3 to 2.6 for the continuous stroke model. Section 2.5 will state concrete criteria for identifying singular points on a stroke’s boundary. These criteria are rooted in the theory of limiting and discriminant envelopes, which will be studied in Sections 2.3 and 2.4. Notably, these two types of envelopes are equivalent for our strokes, as will be proven in Section 2.4.1. For strokes with a constant radius function, the singular boundary points are characterized by the properties of the limiting envelope (Section 2.3.1). The local structure of the discriminant envelope gives important insights into singular boundary points for non-constant radius functions. Sections 2.4.2 and 2.4.3 will provide a theoretical definition of singular boundary points for discriminant envelopes. In addition, Section 2.6 will examine the theory of so-called unfoldings and catastrophe theory, with the intention of classifying the singular boundary points in detail. The results will be used in Section 4.3.2 to state the approach for detecting singular boundary points in the discrete stroke model. Finally, Section 5.4.2 will contain our ideas on how to include non-univalent strokes with singular boundary points in the conformal mapping algorithm.

Chapter 6 will present an extension of our algorithm from Chapter 5 that enables adaptive ornamental design, i.e., design with different levels of detail depending on the appearance of the stroke. The adaptation of the algorithm will be detailed in Section 6.1, with examples of floral and braid designs provided in Section 6.2.

All figures in this thesis were created with Cinderella [RGK23a], CindyJS [RGK23b] or Mathematica [WR23] if not stated otherwise. The code for the proposed algorithms was written in CindyJS. All code files can be found in [Pol23b].

2 | Continuous Model

The continuous stroke model is essential to comprehend the structure and theory behind the target domain of the conformal map we want to compute

In this chapter, we will first define the properties of the curve γ and the radius function r on which the definition of a stroke is based (Section 2.1). We will establish a criterion for the derivative of the radius, which has to be fulfilled in order to guarantee well-defined strokes by intersecting circles. As an interesting addition to the topic, we will connect this characteristic to aerodynamics.

In Section 2.2, the stroke will be considered as a family of circles. We define the boundary of a stroke as the two curves defining the envelope of the family, connected by two arcs from the first and last circles. We will provide three distinct definitions for the stroke envelope: the limiting envelope, the discriminant envelope, and the tangential envelope. We will show in Section 2.4.1 that the first and the second definition of envelopes are equivalent in our context and, furthermore, that the last one is equivalent for regular points.

In Section 2.3, we will determine equations for all points in the limiting envelope and study the properties of these point-sets separately for constant and non-constant radius functions. For strokes with a constant radius function, the envelope consists of the two parallels to the curve γ , which are regular as long as they do not have a common point with the curve's evolute, i.e., the locus of the centers of curvature of γ . For strokes with non-constant radius function, we will specify a condition on the curvature of γ to distinguish regular from non-regular envelope points.

In Section 2.4, we will deduce the definition of the discriminant envelope of a family of circles. We will show that the discriminant and the limiting envelope coincide for our strokes. To study singular boundary points from the perspective of the discriminant envelope, we will consider a related manifold in \mathbb{R}^3 , whose projection onto the plane is a stroke. The envelope is a subset of the projection of the critical points of this manifold, and we will derive that the envelope is regular if the second derivative of the circle function with respect to the curve parameter t is non-zero.

Section 2.5 provides a criterion for the curvature of γ that determines singular envelope points, taking into account that the discriminant and limiting envelopes are identical.

Finally, in Section 2.6, we will use the theory of so-called unfoldings and A_k singularities to demonstrate that the discriminant envelopes of our strokes around non-regular points are diffeomorphic to the discriminant set of the function $g(t) = t^3$. However, this only applies to strokes without self-intersections and to A_2 singularities, for which it will be shown that the boundary of the stroke has a cusp. Furthermore, we will study the envelopes in connection with swallowtail surfaces from catastrophe theory.

2.1 Definition of a stroke

In this section, we provide a formal definition for continuous strokes. The stroke model relies on the curve drawn by the user. The continuously drawn curve

$$\begin{aligned} \gamma: I &\rightarrow \mathbb{R}^2 \\ t &\mapsto \begin{pmatrix} \gamma_x(t) \\ \gamma_y(t) \end{pmatrix} \end{aligned}$$

for a closed interval $I \subset \mathbb{R}$ can be considered as the “backbone” of the stroke model. It is continuously differentiable, regular, i.e., $\gamma'(t) \neq 0$, and not closed.

Additionally, the model includes a so-called *radius function* $r: I \rightarrow \mathbb{R} \setminus \{0\}$. We restrict the value of $r(t)$ to be positive in this thesis, but it makes no difference as long as it is non-zero, as we will see shortly. The radius function is constant $r_c \in \mathbb{R} \setminus \{0\}$ when the user draws with a mouse on a computer or with a finger on a tablet. If a user draws on a tablet using a digital pen capable of registering pressure and possibly also tilt, the radius function is non-constant and continuous. Section 5.1 will define how the radius function is derived from the pressure and tilt input. In this section, for simplicity, we will only name pressure sensitivity as an influencing parameter of the radius function. However, additional information can be gathered from modern digital pens that could be used to derive a radius function, such as the speed of the drawing or the orientation of the pen tip with respect to the drawing surface.

Together, the curve γ and the radius function r define the stroke as the union of the circular disks positioned around the curve point $\gamma(t)$ at each $t \in I$ with radius $r(t)$. The domain covered by these circular disks defines the stroke.

Definition 2.1.1 (Stroke)

The stroke s based on the regular curve $\gamma: I \rightarrow \mathbb{R}^2$ and a continuous radius function $r: I \rightarrow \mathbb{R}_{>0}$ is given by

$$s = \bigcup_{t \in I} \mathcal{D}(t) \quad \text{with} \quad \mathcal{D}(t) = \{(x, y) \in \mathbb{R}^2: \|(x, y)^T - \gamma(t)\|^2 \leq r(t)^2\}.$$

The *interior* of the stroke s is defined as the set of all points located within the unbounded disks $\mathcal{D}(t) \setminus \mathcal{C}(t)$ for the circles $\mathcal{C}(t) = \{(x, y) \in \mathbb{R}^2: \|(x, y)^T - \gamma(t)\|^2 - r(t)^2 = 0\}$ and for all $t \in I$:

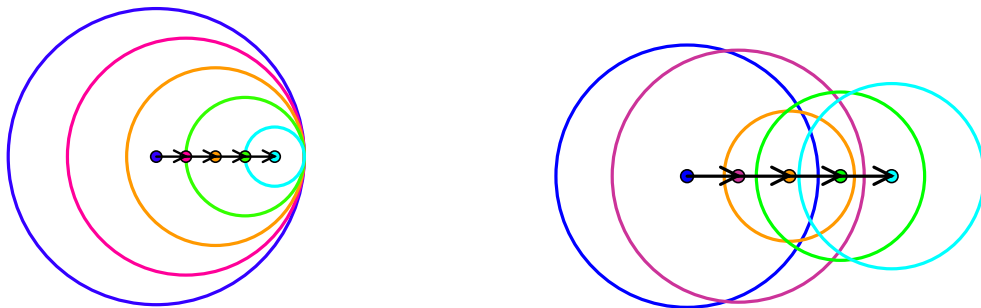
$$\bigcup_{t \in I} \{(x, y) \in \mathbb{R}^2: \|(x, y)^T - \gamma(t)\|^2 < r(t)^2\}. \quad (2.1)$$

The definition of the stroke’s boundary is crucial for understanding the stroke model and for constructing conformal maps from a tiled strip to a stroke. However, the definition is not that simple and will be discussed in detail in the next sections.

As previously stated, it is not important whether the radius values are negative or positive. In either case, the same circle \mathcal{C} or disk \mathcal{D} is defined. However, if the radius were zero, the circle would degenerate to a point, and the stroke would be reduced to the curve

point on γ . We exclude this degenerate case from the definition of the radius function because, in the implementation of the stroke, the radius function is set to zero when the user stops drawing (see Section 5.1). Since the continuous radius function r never equals zero, it will either be positive everywhere or negative everywhere, but it will not change its sign for any $t \in I$. Henceforth, we assume that the radius function is positive for all $t \in I$. We will mention the negative case only if it is not obvious that it does not change the given definition, result, or argumentation.

To keep our model under control, we need to exclude circular disks that are nested, or large disks that cover the rest of the stroke. Figure 2.1 shows examples of circles around curve points $\gamma(t)$ that are positioned in a way that we want to exclude for our stroke model. To simplify the illustrations, we will only show the circles bounding the disks that define a stroke in the following.



(a) $\gamma(t) = (t, 0)^T$ and $r(t) = 1 - t$ for $t \in (0, 1)$ (b) the third circle is contained in the second

Figure 2.1: Counterexamples of subsequent circles along a curve.

To prevent such cases, we need two infinitesimally close circles to intersect in two different real intersection points.

Definition 2.1.2 (Infinitesimally close circles)

A circle $\mathcal{C}(t_0)$ is considered infinitesimally close to another circle $\mathcal{C}(t)$ if the parameter t_0 is within the ϵ -neighborhood of t . This means that for arbitrarily small, positive ϵ , it holds that t_0 is in the interval $(t - \epsilon, t + \epsilon)$.

Proposition 2.1.3 (Intersection of subsequent circles)

If the radius function r and the curve γ satisfy the condition

$$r'(t)^2 < \|\gamma'(t)\|^2, \quad (2.2)$$

then two infinitesimally close circles along the curve γ with radii given by r intersect in two different real intersection points.

Proof: Let $\mathcal{C}(t_0)$ be a circle with center $\gamma(t_0) = (\gamma_x(t_0), \gamma_y(t_0))$ and radius $r(t_0)$. Let $\mathcal{C}(t_1)$ be a second circle defined analogously for $t_1 = t_0 + \epsilon^*$, where $\epsilon^* > 0$ is infinitesimally close to zero. These two circles have two different real intersection points if and only if

$$(r(t_1) - r(t_0))^2 < \|\gamma(t_1) - \gamma(t_0)\|^2 < (r(t_1) + r(t_0))^2 \quad (2.3)$$

[BGT20, Remark 2]. Since we consider the behavior of infinitesimally close circles, we can neglect the right inequality, since this case only occurs for two circles with a distance of at least $|r(t_1) + r(t_0)|$ between the circle centers $\gamma(t_1)$ and $\gamma(t_0)$.

Hence, we show that $(r(t_1) - r(t_0))^2 < \|\gamma(t_1) - \gamma(t_0)\|^2$ follows from $r'(t)^2 < \|\gamma'(t)\|^2$ for infinitesimally close t_0 and $t_1 = t_0 + \epsilon^*$ [KS23].

$$\begin{aligned} r'(t)^2 < \|\gamma'(t)\|^2 &\Leftrightarrow 0 < \|\gamma'(t)\|^2 - r'(t)^2 \\ \Leftrightarrow 0 < \left\| \lim_{\delta \rightarrow 0} \frac{\gamma(t + \delta) - \gamma(t)}{\delta} \right\|^2 - \left(\lim_{\delta \rightarrow 0} \frac{r(t + \delta) - r(t)}{\delta} \right)^2 \\ &\stackrel{\diamond}{=} \lim_{\delta \rightarrow 0} \left(\frac{\|\gamma(t + \delta) - \gamma(t)\|}{\delta} \right)^2 - \lim_{\delta \rightarrow 0} \left(\frac{r(t + \delta) - r(t)}{\delta} \right)^2 \end{aligned} \quad (2.4)$$

$$\stackrel{\bullet}{=} \lim_{\delta \rightarrow 0} \left(\left(\frac{\|\gamma(t + \delta) - \gamma(t)\|}{\delta} \right)^2 - \left(\frac{r(t + \delta) - r(t)}{\delta} \right)^2 \right) \quad (2.5)$$

Equality \diamond holds true since the norm and the quadratic function are continuous. The equality \bullet holds true because both limits exist in (2.4). Therefore, the limit of the subtracted sequence elements in (2.5) is equivalent to the difference of the individual limits. Hence, we obtain a convergent sequence in (2.5) with limit $a := (\|\gamma'(t)\|^2 - r'(t)^2)$ and elements $a_n := \left(\frac{\|\gamma(t + \delta_n) - \gamma(t)\|}{\delta_n} \right)^2 - \left(\frac{r(t + \delta_n) - r(t)}{\delta_n} \right)^2$ where $\delta_n \xrightarrow{n \rightarrow \infty} 0$. For this convergent sequence, it holds

$$\forall \epsilon > 0 \exists N \in \mathbb{N} \text{ such that } \forall n > N : |a_n - a| < \epsilon.$$

Since this holds for all $\epsilon > 0$, we select $0 < \epsilon < a$. If $a_n < 0$, it holds for each n that $|a_n - a| = |-(|a_n| + a)| = |a_n| + a > \epsilon$, which contradicts the assumption. Thus, it holds that $a_n > 0$ for all $n > N$. Hence, for a sufficiently small ϵ^* , it holds

$$\begin{aligned} 0 < a_n &= \left(\frac{\|\gamma(t + \epsilon^*) - \gamma(t)\|}{\epsilon^*} \right)^2 - \left(\frac{r(t + \epsilon^*) - r(t)}{\epsilon^*} \right)^2 \\ &\Leftrightarrow \left(\frac{r(t + \epsilon^*) - r(t)}{\epsilon^*} \right)^2 < \left(\frac{\|\gamma(t + \epsilon^*) - \gamma(t)\|}{\epsilon^*} \right)^2 \\ &\stackrel{\circ}{\Leftrightarrow} \left(\frac{r(t_1) - r(t_0)}{t_1 - t_0} \right)^2 < \left(\frac{\|\gamma(t_1) - \gamma(t_0)\|}{t_1 - t_0} \right)^2 \\ &\Leftrightarrow (r(t_1) - r(t_0))^2 < \|\gamma(t_1) - \gamma(t_0)\|^2. \end{aligned}$$

For the equivalence \circ , we substituted t with $t_0 = t$ and $t_1 = t + \epsilon^*$. The last equivalence applies since $t_1 - t_0 = \epsilon^* > 0$. This concludes the proof. \square

If Condition (2.2) is not satisfied for a curve γ and a radius function r , it is possible for subsequent circles to intersect in less than two real intersection points. Hence, to avoid nested or tangential neighboring circles along the curve, Condition (2.2) has to hold for γ and r of our strokes.

Note 2.1.4 (Pencils of circles)

No matter how two circles are positioned with respect to each other, there is a unique pencil of circles containing them. Following Coxeter [Cox69, Chapter 6.5], two intersecting circles span an elliptic pencil of coaxial circles (see Figure 2.2a), two circles that do not intersect are contained in a hyperbolic pencil of coaxial circles (see Figure 2.2b), and two tangent circles belong to a parabolic pencil of coaxial circles (see Figure 2.2c). The radical axis is the line that connects the real intersection points in the elliptic pencil, the complex intersection points in the hyperbolic pencil, and which is the tangent to the circles in the parabolic case. It is depicted in red in all three cases. It would be interesting to study how the model would behave and how the algorithms of this work would have to be adapted if the Condition (2.2) did not guarantee two real intersection points between two consecutive circles, but if complex intersection points had to be taken into account.

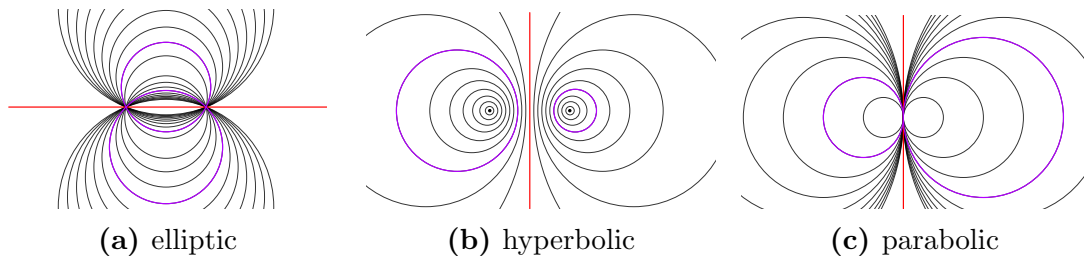


Figure 2.2: Pencils of circles defined by two circles.

Note 2.1.5 ((Super-)sonic flow)

The left example in Figure 2.1 reveals other related topics known in the context of airplanes: shock waves, compressible flows, and (super-)sonic speed. As explained by Bölkow [Böl13, Chapter 2], pressure waves expand uniformly in all directions from a stationary airplane at the speed of sound of 1236 km/h (see Figure 2.3a). When the airplane is moving at subsonic speed, i.e., slower than 1236 km/h, the pressure waves still expand in all directions, but they are no longer uniformly distributed around the airplane (see Figure 2.3b). If an airplane moves at the exact speed of sound, the waves move with the same speed as the airplane, and the so-called sonic barrier occurs, as shown by the blue line tangent to the wavefront in Figure 2.3c. Thus, the aircraft cannot be heard on the other side of the sonic barrier in front of the airplane. If the airplane moves even faster than the sound, it leaves its waves behind, as in Figure 2.3d. The cone in which the waves expand is known as the Mach cone, named after physicist Ernst Mach (1838-1916). We observe that the wave fronts intersect at two points when projected to the plane, and the projected Mach cone is bordered by two blue lines. As the previous citation [Böl13] is in German, we provide two alternative sources in English. For an overview of fluid dynamics, see [Per13], which includes a discussion of supersonic flow in Chapter 9. For an introduction to the fundamentals of aerodynamics, see [And16], which shows the subsonic and supersonic flow cases in Figure 9.4.

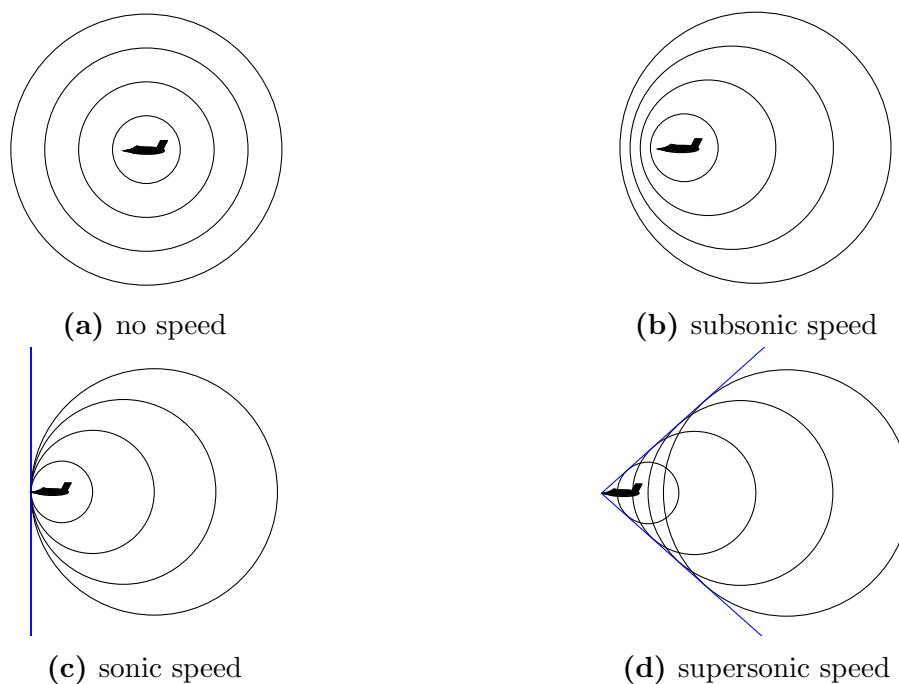


Figure 2.3: The pressure waves of an airplane.

To connect the previous note about supersonic flow back to our strokes, we notice that the chain of circles in Figure 2.3d is a valid chain for a stroke along a straight curve γ , since two neighboring circles intersect. Equation (2.2) requires the pen to move faster than the radius grows. So we have a “supersonic pen” with respect to the radius of the circles. The two lines that border the circles in Figure 2.3d naturally appear as the lines that are tangent to all of these circles. We will examine the boundaries of strokes in detail in the next sections.

In summary, there are few ingredients for our stroke model. The stroke is defined as the union of the circular disks with center $\gamma(t)$ and radius $r(t)$ for all $t \in I$. The curve γ is regular and the radius function r must satisfy Equation (2.2).

We will now introduce some examples of strokes defined by explicit curves γ and radius functions r , which we will refer to throughout this thesis. These examples provide useful benchmarks, each one being a prototype of a specific aspect of our stroke model. They will be used to perform reproducible tests of our algorithm presented in Chapter 5. Since the provided examples are given by analytic functions, we are able to compute certain parts of the strokes explicitly, which allows us to study the properties of their boundaries and to compare the conformality of our calculations with existing conformal maps. We specify the size of the drawing surface in order to have a reference for the parameterization of γ and r : it covers the rectangular region of the xy -plane with $0 \leq x \leq 1024$ and $0 \leq y \leq 512$.

The file *BenchmarkExamples_OrnaStrokes_LP.html* available in [Pol23b] provides an applet that applies our algorithm presented in Chapter 5 to the upcoming benchmark strokes.

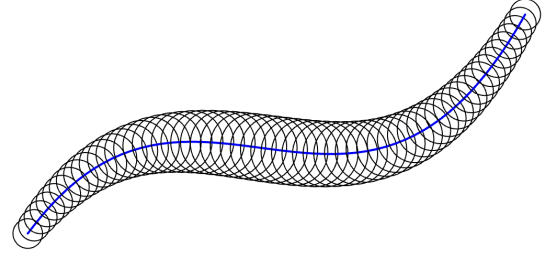
Example 2.1.6 (Typical stroke without exceptions)

For a standard stroke with no exceptional behavior, we define the drawn curve to be

$$\gamma(t) = \begin{pmatrix} 100 \cdot (t - 2) \\ 10 \cdot (t - 1)(t - 3)^2 \end{pmatrix} + \begin{pmatrix} 350 \\ 250 \end{pmatrix}$$

and use the radius function

$$r(t) = 40 \cdot \left(\frac{7}{10} \cdot \cos \left(\frac{(t - 2)}{2} - \frac{1}{4} \right) + \frac{1}{10} \right)$$



for $t \in [0, 5]$. This example will serve as standard example.

Example 2.1.7 (Half annulus)

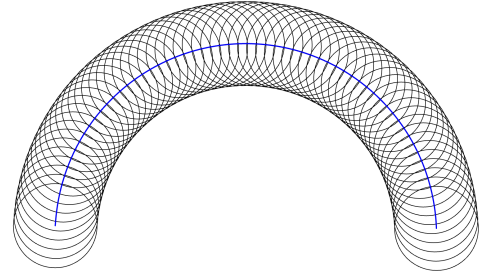
A stroke with the shape of a half annulus will be interesting to compare the conformal modulus between a known exact conformal map and the result of our algorithm.

For this, the curve is defined by

$$\gamma(t) = 300 \cdot \left(\frac{1}{2} \cdot \left(\frac{1}{\rho} + \rho \right) \right) \begin{pmatrix} \cos(\pi - t) \\ \sin(\pi - t) \end{pmatrix} + \begin{pmatrix} 500 \\ 100 \end{pmatrix}$$

and the radius function is constant

$$r(t) \equiv 300 \cdot \frac{1}{2} \left(\frac{1}{\rho} - \rho \right)$$



for $t \in [0, \pi]$ and $\rho = 0.8$.

The smaller parameter $\rho \in (0, 1)$ is, the larger the radius and the wider the half annular region covered by the stroke.

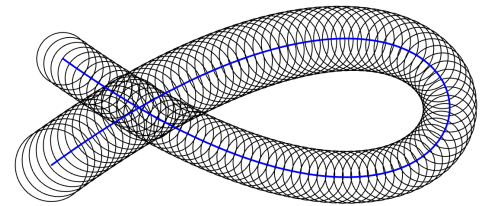
Example 2.1.8 (Self-intersecting stroke)

A self-intersecting, i.e., non-univalent stroke is obtained by the curve

$$\gamma(t) = 40 \cdot \begin{pmatrix} 1 - 3(t - 2)^2 \\ (t - 2)(3 - (t - 2)^2) \end{pmatrix} + \begin{pmatrix} 600 \\ 250 \end{pmatrix}$$

and the radius function

$$r(t) = 60 \cdot \left(\frac{1}{5} \cdot \cos \left(-\frac{t}{2} \right)^4 + \frac{1}{2} \right)$$



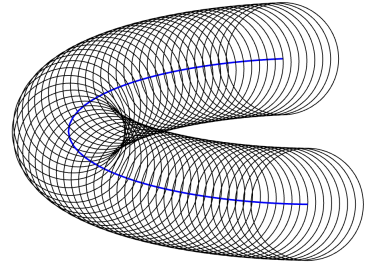
for $t \in [0, 4]$.

Example 2.1.9 (Part of an ellipse)

To study the behavior of strokes with singular boundary points folding over themselves, we analyze the part of an ellipse given by

$$\gamma(t) = 300 \cdot \begin{pmatrix} a \cos \left(t + \frac{\pi}{2} \right) \\ b \sin \left(t + \frac{\pi}{2} \right) \end{pmatrix} + \begin{pmatrix} 500 \\ 250 \end{pmatrix}$$

for $t \in [0, \pi]$, $a = \cosh(0.3)$, $b = \sinh(0.3)$ and with constant radius function of for example $r(t) \equiv 70$.



Example 2.1.10 (Stroke with narrow turn)

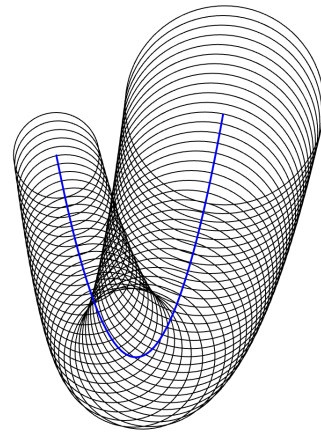
Another example of a stroke that folds over itself is provided by the curve

$$\gamma(t) = 100 \cdot \begin{pmatrix} (t - 1) \\ 3 \cdot (t - 1)^2 \end{pmatrix} + \begin{pmatrix} 400 \\ 100 \end{pmatrix}$$

with non-constant radius function

$$r(t) = 140 \cdot \cos \left(\frac{t}{4} - 1 \right)^2$$

for $t \in [0, 2]$.



2.2 Envelopes of a family of curves

To analyze the characteristics of the boundary of a stroke in our model, we need to study the properties of the set of circles that define the stroke. For this, we give a general definition of a family of curves in the plane. In this context, a curve is no longer given as a parameterization, but as the solution of an equation.

Definition 2.2.1 (Family of curves)

A family of curves in \mathbb{R}^2 is a collection of curves $F(t_1, \dots, t_m, x, y) = 0$, depending on the free parameters (t_1, \dots, t_m) . For any parameter in the m -dimensional parameter space, the equation $F(t_1, \dots, t_m, x, y) = 0$ represents a curve of the family, called a family member.

According to this definition and following Bickel et al. [BGT20, Section 4], a stroke is defined by a family of circles which we call \mathcal{F} . Each circle of the family is given as the set of points (x, y) which satisfy

$$F(t, x, y) = \|(x, y)^T - \gamma(t)\|^2 - r(t)^2 = 0 \tag{2.6}$$

for curve points $\gamma(t) = (\gamma_x(t), \gamma_y(t))^T$ with $t \in I$, and for a radius function r . We denote a specific circle in the family associated with the parameter $t \in I$ by

$$\mathcal{C}(t) = \{(x, y) \in \mathbb{R}^2 \mid F(t, x, y) = 0\} \tag{2.7}$$

which corresponds to the previous definition of the circles given after Definition 2.1.1. Zero is a regular value of $F(t, x, y)$ because

$$\begin{aligned} \frac{\partial F}{\partial x}(t, x, y) &= 2(x - \gamma_x(t)) = 0 \Leftrightarrow x = \gamma_x(t) \\ \frac{\partial F}{\partial y}(t, x, y) &= 2(y - \gamma_y(t)) = 0 \Leftrightarrow y = \gamma_y(t). \end{aligned} \tag{2.8}$$

The partial derivatives in (2.8) are equal to zero if and only if $(x, y) = \gamma(t)$. However, this cannot happen for $F(t, x, y) = 0$, since the points (x, y) are defined to be located on the circle $\mathcal{C}(t)$ with non-zero radius $r(t)$ around $\gamma(t)$.

A family of curves has a natural associated envelope. Loosely speaking, the envelope of a family of curves consists of the curve(s) that surround the members of the family, forming a touching envelope around them. Looking at Figure 2.4, one may easily guess that the circle, the horizontal line, and the two intersecting lines each define the envelope of the depicted families of curves. However, properly defining the envelope for a family of curves is not as easy as it seems.

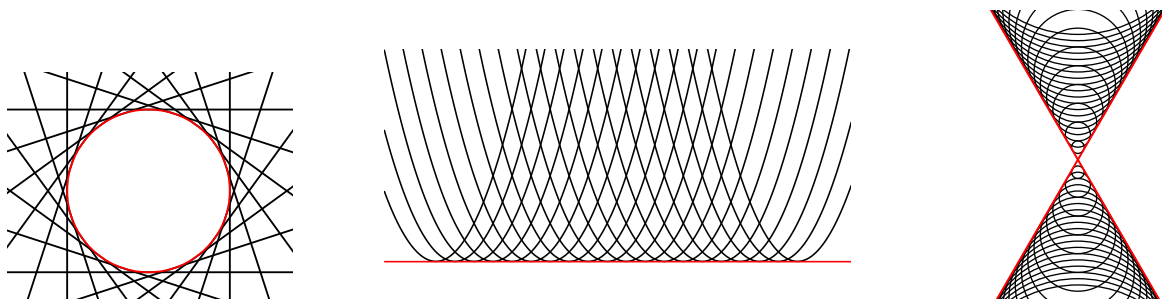


Figure 2.4: Examples for envelopes of families of different curves.

According to Bickel et al. [BGT20] and Bruce and Giblin [BG92], there exist three different definitions of the envelope of a family of curves. Since our families of circles only have one parameter, we will limit the dimension of the parameter space to $m = 1$ in the subsequent definitions, but they also apply to $t \in \mathbb{R}^m$ for $m > 1$.

Definition 2.2.2 (Discriminant envelope)

The discriminant envelope of a family of curves is defined as the set of points E_d :

$$E_d = \left\{ (x, y) \in \mathbb{R}^2 : \exists t \in \mathbb{R} \text{ with } F(t, x, y) = 0 \text{ and } \frac{\partial F}{\partial t}(t, x, y) = 0 \right\}.$$

The discriminant envelope is the largest set of envelope points. It provides an analytical view on the family of curves.

Definition 2.2.3 (Limiting envelope)

The limiting envelope E_ℓ of a family of curves is defined as the set of limiting intersection points of neighboring curves \mathcal{C}_t and $\mathcal{C}_{t+\epsilon}$ for $\epsilon \rightarrow 0$.

Bruce and Giblin show that all points in the limiting envelope of a family of curves are contained in the discriminant envelope of the same family of curves [BG92, 5.8]. For our strokes, the limiting envelope and the discriminant envelope even contain exactly the same points, which we will prove in Section 2.4.1.

Definition 2.2.4 (Tangential envelope)

The envelope E_t of a family of curves is the union of smooth curves that are tangent to a family member $\mathcal{C}(t) = \{(x, y) \in \mathbb{R}^2 : F(t, x, y) = 0\}$ for each t . So, for any tangent curve $\varphi: J \subseteq \mathbb{R} \rightarrow \mathbb{R}^2$, it holds for all $t \in J$ and $\varphi(t) = (\varphi_x(t), \varphi_y(t))^T$ that

- i) the point $\varphi(t)$ is located on $\mathcal{C}(t)$, i.e., $F(t, \varphi_x(t), \varphi_y(t)) = 0$,
- ii) φ and $\mathcal{C}(t)$ share the same tangent at $\varphi(t)$.

Also, E_t is generally contained in but not equal to the discriminant envelope E_d . Overall, we can conclude for all three envelopes that

$$E_t \subseteq E_d \quad \text{and} \quad E_\ell \subseteq E_d. \quad (2.9)$$

We will see in Proposition 2.4.7 that the tangential envelope matches the discriminant envelope for our strokes if the envelope has no singular point.

Depending on the context, it may be advantageous to use one of the three definitions. For a geometric understanding of the envelope of a family of curves, it may be convenient to think of the curve tangent to all family members. For example, the envelopes in Figure 2.4 were immediately recognizable as tangents to all curves. For analytical computations, the discriminant envelope may be the most appropriate choice. We will, for example, determine interesting properties of the boundary of the strokes by analyzing their discriminant envelopes. In some cases, however, the best way to handle the envelope is to calculate the intersection points of neighboring curves of the family. We will see that the set of points in E_ℓ can be explicitly parameterized, which makes it easy to draw the envelope.

We will show in Section 2.4.1 that our families of circles have equal discriminant envelope E_d and limiting envelope E_ℓ . Thus, we study the properties of the envelope of a stroke by analyzing both E_ℓ and E_d for families of circles.

Before we discuss the details, we define the boundary of a stroke as the union of its two envelope curves and two circular arcs. These circular arcs are subsets of the first and last circles $\mathcal{C}(0)$ and $\mathcal{C}(T)$ of a stroke defined on the closed interval $I = [0, T]$. The two circles each contain two endpoints of the two envelope curves which will be shown in Section 2.3. Hence, the circular arcs connect these two envelope points. Figure 2.5 shows the boundary of the strokes of the Benchmark Examples 2.1.6 and 2.1.8. The envelope curves are shown in red, the circular arcs in blue, and the displayed points are the end points of the envelope curves that border the circular arcs. It is evident from the right picture that the envelope curves together with the two circular arcs do not define the metric boundary of the stroke. If the boundary of a stroke s was defined as its metric boundary, i.e., as the closure of $s = \bigcup_{t \in I} \mathcal{D}(t)$ minus its interior [Kai23, p.69] as defined in Equation (2.1), then the parts of the envelope curves that border the parallelogram-like shape in Figure 2.5 on the right would not be part of its boundary, since they are

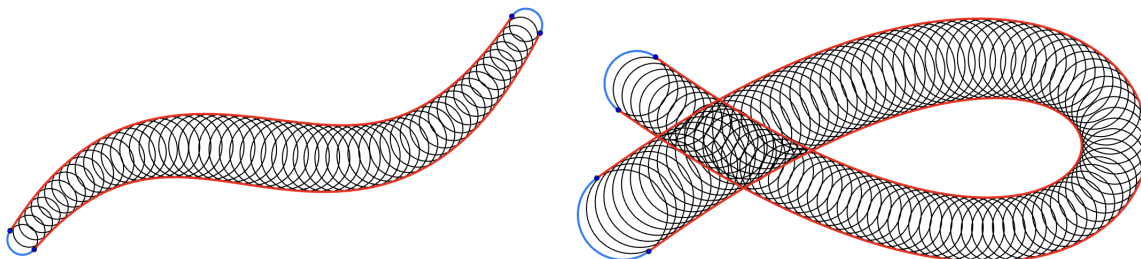


Figure 2.5: The boundary of a stroke consisting of the envelope curves and circular arcs.

contained in the stroke's interior. However, the boundary of a stroke must contain the complete envelope curves so that we can correctly map the ornamental strip to the stroke.

Definition 2.2.5 (Boundary of a stroke) [BGT20, Theorem 2]

Given a stroke s for the interval $I = [0, T]$, a regular curve $\gamma: I \rightarrow \mathbb{R}^2$, and a continuous radius function $r: I \rightarrow \mathbb{R}_{>0}$, for which it holds that $r'(t)^2 < \|\gamma'(t)\|^2$ as in Proposition 2.1.3. Then the boundary of s is denoted by ∂s . It is contained in the union of the stroke's envelope and the circles \mathcal{C}_0 and \mathcal{C}_T :

$$\partial s \subseteq E_d \cup \mathcal{C}_0 \cup \mathcal{C}_T.$$

Furthermore, the discriminant envelope is entirely contained in the boundary: $E_d \subseteq \partial s$.

2.3 Limiting envelope of a family of circles

To study the structure of the boundary of a stroke, we first examine the points in the limiting envelope E_ℓ of the corresponding family of circles. We partially follow the argumentation of Bickel et al. [BGT20, Section 4].

According to the definition of E_ℓ that it consists of the intersection points of two infinitesimally close circles, we consider two circles $\mathcal{C}(t)$ and $\mathcal{C}(t_0)$ for $t \neq t_0 \in I$ with $t_0 \in (t - \epsilon, t + \epsilon)$ for an arbitrary small $\epsilon > 0$ as in Definition 2.1.2. The circles have centers $\gamma(t) = (\gamma_x(t), \gamma_y(t))^T$ and $\gamma(t_0) = (\gamma_x(t_0), \gamma_y(t_0))^T$, as well as the radii $r(t)$ and $r(t_0)$, respectively.

As we already know from Equation (2.3), two circles have two real intersection points if and only if

$$(r(t) - r(t_0))^2 \leq \|\gamma(t) - \gamma(t_0)\|^2 \leq (r(t) + r(t_0))^2.$$

For two infinitesimally close circles such as $\mathcal{C}(t)$ and $\mathcal{C}(t_0)$, the right inequality holds because their centers are always closer together than $|r(t) + r(t_0)|$. Since we excluded in Proposition 2.1.3 that $\mathcal{C}(t)$ and $\mathcal{C}(t_0)$ are internally tangent to each other, they will intersect in two different real points p_1 and p_2 . For the nomenclature used in the following calculations of p_1 and p_2 , see Figure 2.6.

To simplify calculations, we define d as the distance between the centers of the circles:

$$d = \|\gamma(t_0) - \gamma(t)\|.$$

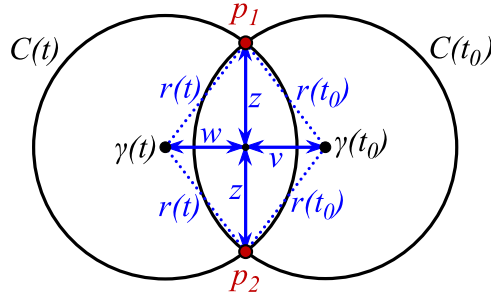


Figure 2.6: Two intersecting circles $\mathcal{C}(t)$ and $\mathcal{C}(t_0)$.

The intersection points p_1 and p_2 are calculated by

$$p_{1/2} = \gamma(t) + w \cdot \frac{\gamma(t_0) - \gamma(t)}{d} \pm z \cdot \frac{(\gamma(t_0) - \gamma(t))^\perp}{d} \quad (2.10)$$

with $v^\perp = (-v_y, v_x)$. For this, we solve the following system of equations for w and z :

$$(I) \quad w^2 + z^2 = r(t)^2 \quad (II) \quad v^2 + z^2 = r(t_0)^2 \quad (III) \quad w + v = d.$$

We derive w as follows:

$$\left. \begin{array}{l} (III)', \quad v = d - w \\ (I)', \quad z^2 = r(t)^2 - w^2 \end{array} \right\} \stackrel{(II)'}{\Rightarrow} (d - w)^2 + r(t)^2 - w^2 = r(t_0)^2 \Rightarrow w = \frac{r(t)^2 - r(t_0)^2 + d^2}{2d}.$$

By equation (I) and w , we get z by

$$z = \sqrt{r(t)^2 - w^2} = \sqrt{r(t)^2 - \frac{(r(t)^2 - r(t_0)^2 + d^2)^2}{4d^2}} = \frac{1}{2d} \sqrt{4d^2 r(t)^2 - (r(t)^2 - r(t_0)^2 + d^2)^2}.$$

The intersection points p_1 and p_2 can be derived by inserting w and z into Equation (2.10)

$$p_{1/2}(t) = \gamma(t) + \frac{r(t)^2 - r(t_0)^2 + d^2}{2d^2} (\gamma(t_0) - \gamma(t)) \pm \frac{1}{2d^2} \sqrt{4d^2 r(t)^2 - (r(t)^2 - r(t_0)^2 + d^2)^2} (\gamma(t_0) - \gamma(t))^\perp.$$

In the limiting envelope, the two circles are centered at infinitesimally close curve points, so we take the limit of $t_0 \rightarrow t$ and use that $\lim_{t_0 \rightarrow t} \frac{\gamma(t_0) - \gamma(t)}{t_0 - t} = \gamma'(t)$:

$$\begin{aligned} \lim_{t_0 \rightarrow t} p_{1/2}(t) &= \lim_{t_0 \rightarrow t} \frac{1}{\frac{(t_0 - t)^2}{(t_0 - t)^2}} p_{1/2}(t) = \\ &= \gamma(t) + \frac{\frac{r(t)^2 - r(t_0)^2 + \|\gamma(t_0) - \gamma(t)\|^2}{(t_0 - t)}}{2\|\gamma'(t)\|^2} \gamma'(t) \pm \frac{1}{2\|\gamma'(t)\|^2} \sqrt{\frac{4d^2 r(t)^2 - (r(t)^2 - r(t_0)^2 + d^2)^2}{(t_0 - t)^2}} (\gamma'(t))^\perp \\ &= \gamma(t) - \frac{r(t)r'(t)}{\|\gamma'(t)\|^2} \gamma'(t) \pm \frac{r(t)}{\|\gamma'(t)\|^2} \sqrt{\|\gamma'(t)\|^2 - r'(t)^2} (\gamma'(t))^\perp \end{aligned}$$

This results in the following set of points defining the limiting envelope of a stroke's family of circles:

$$E_\ell = \left\{ P_\pm(t) = \gamma(t) - \frac{r(t)r'(t)}{\|\gamma'(t)\|^2} \gamma'(t) \pm \frac{r(t)}{\|\gamma'(t)\|^2} \sqrt{\|\gamma'(t)\|^2 - r'(t)^2} (\gamma'(t))^\perp, t \in I \right\}. \quad (2.11)$$

If curve γ is arc-length parameterized, i.e., the norm of its derivative is one, $\|\gamma'(t)\| = 1$, the limiting envelope is given by:

$$E_\ell = \left\{ P_\pm(t) = \gamma(t) - r(t)r'(t)T(t) \pm r(t)\sqrt{1 - r'(t)^2}N(t), t \in I \right\} \quad (2.12)$$

for the unit length tangent $T(t) = \gamma'(t)$ and the unit length normal vector $N(t) = (\gamma'(t))^\perp$ with the property that $\langle T(t), N(t) \rangle = 0$. The set of points in Equation (2.11) is more general, but E_ℓ in Equation (2.12) sometimes provides a better geometric insight.

All points $P_\pm(t)$ of the limiting envelope E_ℓ are reached from the curve point $\gamma(t)$ by moving by $(-r(t)r'(t))$ in the direction of the tangent of γ and by $(\pm r(t)\sqrt{\|\gamma'(t)\|^2 - r'(t)^2})$ along the perpendicular normal. If the radius increases in a subinterval of I , its derivative is positive and the intersection points between a circle around $\gamma(t)$ and its successor will be “behind” $\gamma(t)$, since the coefficient of $T(t)$ is negative. If the radius decreases, the opposite is true and the intersection points between circles and their successors around $\gamma(t)$ lie “ahead of” the curve point $\gamma(t)$. Both of these cases are illustrated in Figure 2.7. If the radius remains constant, its derivative is zero and the intersection points have only a distance in the direction of the normal to the curve. This case will be covered in the following Section 2.3.1.



Figure 2.7: A change in the radius determines the direction of intersection.

Instead of treating E_ℓ as a set of points $P_\pm(t)$ for all $t \in I$, we can also consider it as a the set of all points lying on two different curves w_\pm . The curve w_+ contains all points P_+ and the curve w_- contains all points P_- :

$$\begin{aligned} w_+(t) &:= \gamma(t) - \frac{r(t)r'(t)}{\|\gamma'(t)\|^2} \gamma'(t) + \frac{r(t)}{\|\gamma'(t)\|^2} \sqrt{\|\gamma'(t)\|^2 - r'(t)^2} (\gamma'(t))^\perp \\ w_-(t) &:= \gamma(t) - \frac{r(t)r'(t)}{\|\gamma'(t)\|^2} \gamma'(t) - \frac{r(t)}{\|\gamma'(t)\|^2} \sqrt{\|\gamma'(t)\|^2 - r'(t)^2} (\gamma'(t))^\perp. \end{aligned} \quad (2.13)$$

Note 2.3.1 Restricting the function r to positive radii does not affect the set E_ℓ . If r were a negative function, it could be expressed as $r(t) = -f(t)$, where $f(t) = |r(t)| > 0$

for all values of t . The derivative can then be written as $r'(t) = -f'(t)$. The second term in w_{\pm} remains unchanged because the negative signs for $r(t)$ and $r'(t)$ cancel each other out. There is no change under the square root in the third summand because the derivative of the radius function is squared. The occurrence of $r(t)$ in front of the square root would actually change the sign of the summand if we used the negative instead of the positive radius function. However, this would only swap the points P_+ and P_- and thus the curves w_+ and w_- . The set of points in E_{ℓ} remains unchanged. Hence, it is not problematic to restrict the radius function r to positive values.

2.3.1 E_{ℓ} for constant radius

In this section, we examine the case of a constant radius function: $r(t) \equiv r_c > 0 \forall t \in I$. Strokes have a constant radius function if they are drawn with a non-pressure-sensitive device. It turns out that there is an interesting connection between envelopes and evolutes.

If the radius function is constant, its derivative is zero, and the limiting envelope reduces to

$$E_{\ell}^{r_c} = \left\{ P_{\pm}(t) = \gamma(t) \pm \frac{r_c}{\|\gamma'(t)\|} (\gamma'(t))^{\perp}, t \in I \right\}.$$

The corresponding two envelope curves are

$$w_+(t) = \gamma(t) + \frac{r_c}{\|\gamma'(t)\|} (\gamma'(t))^{\perp} \quad \text{and} \quad w_-(t) = \gamma(t) - \frac{r_c}{\|\gamma'(t)\|} (\gamma'(t))^{\perp}$$

for $t \in I$. These are the parallel curves to curve γ at a constant distance of r_c and $-r_c$ in the direction of the normal vector $(\gamma'(t))^{\perp}$, i.e., the tangent $\gamma'(t)$ rotated by $+\frac{\pi}{2}$.

Definition 2.3.2 (Parallels to a regular curve)

Given the regular curve $\gamma: I \rightarrow \mathbb{R}^2$, its parallel $\gamma_{\parallel r_c}: I \rightarrow \mathbb{R}^2$ is defined as

$$\gamma_{\parallel r_c}(t) = \gamma(t) + \frac{r_c}{\|\gamma'(t)\|} (\gamma'(t))^{\perp}$$

for $t \in I$ and $r_c \in \mathbb{R} \setminus \{0\}$.

To study the regularity of the parallel, we calculate its derivative:

$$\gamma'_{\parallel r_c}(t) = \gamma'(t) + \frac{r_c(\gamma''(t))^{\perp}}{\|\gamma'(t)\|} + \frac{r_c\gamma'(t)(\gamma'_x(t)\gamma''_x(t) + \gamma'_y(t)\gamma''_y(t))}{\|\gamma'(t)\|^3} = \gamma'(t)(1 - r_c \cdot \kappa(t))$$

for $\kappa(t) = \frac{\gamma'_x(t)\gamma''_y(t) - \gamma'_y(t)\gamma''_x(t)}{\|\gamma'(t)\|^3}$ the curvature of the curve $\gamma(t) = (\gamma_x(t), \gamma_y(t))^T$ and $t \in I$.

In accordance with Rutter [Rut18, Corollary 12.4], it can be concluded that the parallel of a regular curve is non-regular in t if and only if.

$$1 - r_c \cdot \kappa(t) = 0 \quad \Leftrightarrow \quad \frac{1}{r_c} = \kappa(t).$$

This relation between the curvature and the radius shows that the parallel at t is non-regular if and only if it passes through the center of curvature of γ at t . The locus of all centers of curvature of a curve γ is a curve called the *evolute*:

Theorem 2.3.3 (and Definition of the evolute) [Rut18, Theorem 12.7]

Let t_0 be a parameter value at which the smooth curve $\gamma: I \rightarrow \mathbb{R}^2$ is regular. Then, its parallel $\gamma_{\parallel r_c}$ and its evolute

$$\nu(t) := \gamma(t) + \frac{1}{\kappa(t)\|\gamma'(t)\|} (\gamma'(t))^\perp$$

meet at t_0 if and only if the parallel is not regular at t_0 .

Thus, for a constant radius r_c , singular points of the envelope occur for all parameters t where the curvature of γ is equal to $\frac{1}{r_c}$ and, hence, the parallel and the evolute of γ meet. If the radius is small, the curvature of γ must be correspondingly large for some parameter $t \in I$ such that singular points on the envelope can actually occur. If the radius is large, a small curvature of curve γ is enough to create singular points on the boundary of the stroke. As an example, Figure 2.8 shows the curve $\gamma(t) = \left(t, \frac{t^2}{4}\right)$, its evolute $\nu(t)$, and its parallel for $r_c = 6$. The parallel has singular points exactly where it meets the evolute.

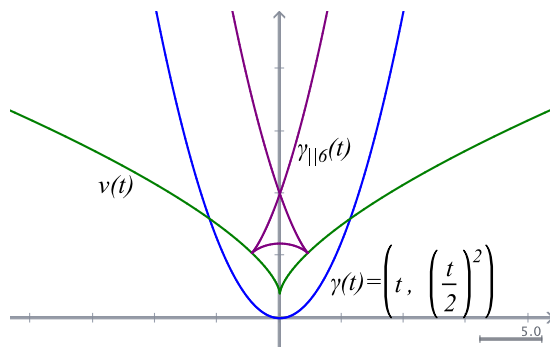


Figure 2.8: A curve γ , its parallel $\gamma_{\parallel r_c}$ for $r_c = 6$ and its evolute ν .

Note 2.3.4 According to Rutter [Rut18, Section 12.1.1], the parallel can also be defined at a parameter t_0 if the curve γ has a non-regular point. For this, the curve's unit tangent should approach some limit as the parameter approaches t_0 . The parallel might be a differentiable curve at t_0 if the singular point of γ is no cusp. If γ has a cusp at t_0 , two separate parallels are needed for the parameters less than and greater than t_0 . However, this detail is not relevant for further analysis because our definition of strokes assumes that the curve γ is regular.

2.3.2 E_ℓ for non-constant radius

If the radius is not constant, the limiting envelope E_ℓ is not reduced to the parallels of curve γ . To gain further understanding of the singular points of w_+ and w_- for non-constant radius functions r , we take their derivatives and interpret the result. For this, we assume that the curve γ is arc-length parameterized, which is why the envelope curves w_\pm are given by Equation (2.12) as

$$w_\pm(t) = \gamma(t) - r(t)r'(t)T(t) \pm r(t)\sqrt{1 - r'(t)^2}N(t).$$

Their derivative is given by

$$\begin{aligned}
 w'_{\pm}(t) &= \gamma'(t) - r'(t)^2 T(t) - r(t)r''(t)T(t) - r(t)r'(t)T'(t) \\
 &\quad \pm r'(t)\sqrt{1-r'(t)^2} \cdot N(t) \pm r(t)\sqrt{1-r'(t)^2} \cdot N'(t) \mp r(t)\frac{r'(t)r''(t)}{\sqrt{1-r'(t)^2}}N(t) \\
 &= T(t) - r'(t)^2 T(t) - r(t)r''(t)T(t) \mp \kappa(t)r(t)\sqrt{1-r'(t)^2} \cdot T(t) \\
 &\quad - \kappa(t)r(t)r'(t)N(t) \pm r'(t)\sqrt{1-r'(t)^2} \cdot N(t) \mp r(t)\frac{r'(t)r''(t)}{\sqrt{1-r'(t)^2}}N(t).
 \end{aligned} \tag{2.14}$$

The second equivalence holds due to the rules of differential geometry for arc-length parameterized γ , that the unit tangent is $T(t) = \gamma'(t)$ and that for the curvature $\kappa(t)$ of γ , $T'(t) = \kappa(t)N(t)$ as well as $N'(t) = -\kappa(t)T(t)$.

The limiting envelope's singular points are given for those $t \in I$ where $w'_{\pm}(t) = 0$. Since the vectors T and N are perpendicular and span the plane, a linear combination of T and N is zero if and only if both coefficients are zero:

$$w'_{\pm}(t) = 0 \Leftrightarrow \begin{cases} 0 = 1 - r'(t)^2 - r(t)r''(t) \mp \kappa(t)r(t)\sqrt{1-r'(t)^2} \\ 0 = -\kappa(t)r(t)r'(t) \pm r'(t)\sqrt{1-r'(t)^2} \mp r(t)\frac{r'(t)r''(t)}{\sqrt{1-r'(t)^2}} \end{cases}$$

If the derivative of the radius function is non-zero, meaning $r'(t) \neq 0$, we can divide the second equation by $r'(t)$ and multiply it by $\sqrt{1-r'(t)^2}$. Then, simple calculations show that the two equations on the right are identical. If the derivative of r with respect to t equals zero, the second equation is always satisfied, leaving only the first equation as a condition.

All in all, the limiting envelope of the arc-length parameterized curve γ has singular points if the curvature satisfies the following condition. We do not divide by zero in the following because the ‘‘supersonic’’ Condition (2.2) guarantees that $r'(t) < 1 = \|\gamma'(t)\|$ and $r(t) > 0$ for the entire stroke.

Proposition 2.3.5 (Singular points in E_{ℓ} for arc-length parameterized curves)

Let γ be an arc-length parameterized curve and let the radius function r be non-constant. Then the limiting envelope of a family of circles \mathcal{F} defined by

$$F(t, x, y) = \|(x, y)^T - \gamma(t)\|^2 - r(t)^2 = 0$$

has a singular point at $w_{\pm}(t)$ if and only if

$$\kappa(t) = \pm \frac{1 - r'(t)^2 - r(t)r''(t)}{r(t)\sqrt{1-r'(t)^2}} \tag{2.15}$$

for the curvature $\kappa(t)$ at the point $\gamma(t)$ for $t \in I$.

To further investigate the properties of singularities on a stroke's boundary, we will examine the definition of the discriminant envelope for a family of circles in the subsequent section. We will revisit Proposition 2.3.5 in Section 2.5.

2.4 Discriminant envelope of a family of circles

The definitions of tangential and limiting envelopes are quite visual: we imagine tangents to a family of curves or intersections of neighboring family members easily. It is harder to imagine the discriminant envelope, which is defined as a set of points (x, y) for which a function $F(t, x, y)$ and its derivative with respect to the parameter t vanish. To illustrate this, we follow the argumentation of Fowler [Fow20, Chapter V §5.10] and deduce where the definition of the discriminant envelope E_d comes from. The nomenclature for the following is visualized in Figure 2.9.

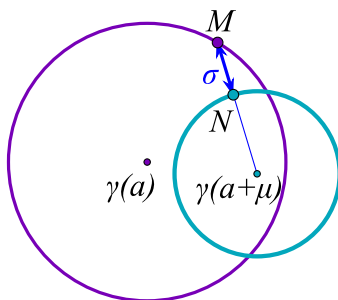


Figure 2.9: Derivation of the definition of E_d .

The family of curves \mathcal{F} defining a stroke consists of circles

$$\mathcal{C}(t) = \{(x, y) \in \mathbb{R}^2 \mid F(t, x, y) = \|(x, y)^T - \gamma(t)\|^2 - r(t)^2 = 0\}$$

which have no singular points for all $t \in [0, T]$.

We choose a point $M = (x_M, y_M)$ on circle $\mathcal{C}(a)$ for a fixed $a \in I$ and define a neighboring circle to $\mathcal{C}(a)$ within a distance of a small $\mu > 0$, denoted by

$$\mathcal{C}(a + \mu) = \{(x, y) \in \mathbb{R}^2 \mid F(a + \mu, x, y) = 0\}.$$

Since none of the members of \mathcal{F} has singular points, both $\mathcal{C}(a)$ and $\mathcal{C}(a + \mu)$ have no singular points. By Taylor expansion in the first variable, the points on the neighboring circle $\mathcal{C}(a + \mu)$ satisfy

$$F(a + \mu, x, y) = F(a, x, y) + \mu \frac{\partial F}{\partial a}(a, x, y) + \mathcal{O}(\mu)^2 = 0. \quad (2.16)$$

We seek the minimum distance σ between the selected point M on $\mathcal{C}(a)$ and the neighboring circle $\mathcal{C}(a + \mu)$, because this gives insight into the change of F w.r.t. t , i.e., insight into $\frac{\partial F}{\partial t}$, which is part of the discriminant envelope definition. In other words, the value of σ indicates the effect a change in the parameter a has on the curve points of one family member with respect to the neighboring one.

Following [Fow20, §4.10], we define $N = (x_N, y_N)$ as the point on $F(a + \mu, x, y) = 0$ that is closest to $M = (x_M, y_M)$. The shortest distance from a point to a circle is found at the point where the line connecting the point to the center of the circle intersects with the circle. The intersection closest to the reference point defines the shortest distance. Any other point on the circle has a greater distance due to the triangle inequality. As

any line passing through the center of a circle is perpendicular to the circle, the vector between points M and N is perpendicular to $\mathcal{C}(a + \mu)$ at point N .

We determine the distance σ by using the direction vector $(v, w) \in \mathbb{R}^2$ of unit length from M to N . By $N = (x_N, y_N) = (x_M + \sigma \cdot v, y_M + \sigma \cdot w)$ with $F(a + \mu, x_N, y_N) = 0$ and by Taylor expansion of $F(a + \mu, x_N, y_N)$ in x_N and y_N , we get

$$0 = F(a + \mu, x_N, y_N) = F(a + \mu, x_M, y_M) + \sigma \left(v \frac{\partial F}{\partial x} + w \frac{\partial F}{\partial y} \right) + \mathcal{O}(\sigma^2). \quad (2.17)$$

We can give formulas for v and w because the vector (v, w) is parallel to the normal at N to the curve $F(a + \mu, x, y) = 0$. One normal vector is the gradient vector $\nabla F = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y} \right)^T$, which is always normal to a level set [Rut18, Thm 4.24]. In our case, $F(a + \mu, x, y) = 0$ is a level set and, thus, we have

$$v = \frac{1}{\sqrt{\frac{\partial F^2}{\partial x} + \frac{\partial F^2}{\partial y}}} \cdot \frac{\partial F}{\partial x}(a + \mu, x, y), \quad w = \frac{1}{\sqrt{\frac{\partial F^2}{\partial x} + \frac{\partial F^2}{\partial y}}} \cdot \frac{\partial F}{\partial y}(a + \mu, x, y).$$

Inserting v and w into Equation (2.17) yields

$$\sigma \sim \frac{-F(a + \mu, x_M, y_M)}{\sqrt{\frac{\partial F^2}{\partial x} + \frac{\partial F^2}{\partial y}}}, \quad (2.18)$$

ignoring the small term of $\mathcal{O}(\sigma^2)$ [Fow20, Thm 4.12].

Returning to the first entry of $F(a + \mu, x, y) = 0$, we are interested in what happens when μ tends to zero, which provides knowledge about infinitesimal change between neighboring members of the family of curves. We recall the Taylor expansion in Equation (2.16) and get

$$\sigma \sim -\frac{F(a, x_M, y_M) + \mu \frac{\partial F}{\partial a}(a, x_M, y_M) + \mathcal{O}(\mu)^2}{\sqrt{\frac{\partial F^2}{\partial x} + \frac{\partial F^2}{\partial y}}}.$$

Since $M = (x_M, y_M)$ lies on $F(a, x, y)$, it holds that $F(a, x_M, y_M) = 0$. Furthermore, all family members are circles and thus have no singular points. Hence, we can rewrite σ for some constant $A \neq 0$ as

$$\sigma \sim -A \left(\mu \frac{\partial F}{\partial a}(a, x_M, y_M) + \mathcal{O}(\mu)^2 \right).$$

When μ tends to zero, the “distance will be of the second or higher order of smallness if and only if” [Fow20, p.59]

$$\frac{\partial F}{\partial a}(a, x_M, y_M) = 0.$$

Those points that undergo minimal changes as they change from one circle to its neighbor are the envelope points that define the discriminant envelope of the family of circles, as in Definition 2.2.2:

$$E_d = \left\{ (x, y) \in \mathbb{R}^2 : \exists t \in \mathbb{R} \text{ with } F(t, x, y) = 0 \text{ and } \frac{\partial F}{\partial t}(t, x, y) = 0 \right\}.$$

Thus, we can imagine the discriminant envelope as consisting of those points on the individual circles that change their position only slightly when moving from one circle to its neighbor.

2.4.1 Equality of discriminant and limiting envelope

We have already stated in (2.9) that the limiting envelope E_ℓ is a subset of the discriminant envelope E_d . Since we have given the points contained in E_ℓ , we can prove that they are contained in E_d , and we will even deduce that for the family of circles of our strokes, both envelopes are the same.

Proposition 2.4.1 ($E_\ell \subseteq E_d$)

The points $P_+(t)$ and $P_-(t)$ belonging to the limiting envelope (2.11) are contained in the discriminant envelope E_d : $E_\ell \subseteq E_d$.

Proof: It has to be shown that $F(t, x, y) = \frac{\partial F}{\partial t}(t, x, y) = 0$ holds for $(x, y) = P_\pm(t)$. Since $P_\pm(t)$ is located on $\mathcal{C}(t)$ by definition of the limiting envelope, $F(t, x, y)|_{P_\pm(t)} = 0$. We also verify that the first derivative of F with respect to t is zero at the points of the limiting envelope:

$$\begin{aligned} \frac{\partial F}{\partial t}(t, x, y) \Big|_{P_\pm(t)} &= \left(-2 \langle (x, y)^T - \gamma(t), \gamma'(t) \rangle - 2r(t)r'(t) \right) \Big|_{P_\pm(t)} = \\ &= -2 \left\langle -\frac{r(t)r'(t)}{\|\gamma'(t)\|^2} \gamma'(t) \pm \frac{r(t)\sqrt{\|\gamma'(t)\|^2 - r'(t)^2}}{\|\gamma'(t)\|^2} \gamma'(t)^\perp, \gamma'(t) \right\rangle - 2r(t)r'(t) \\ &= 2 \left(\frac{r(t)r'(t)\|\gamma'(t)\|^2}{\|\gamma'(t)\|^2} - r(t)r'(t) \right) = \\ &= 2 \cdot r(t)r'(t) \left(\frac{\|\gamma'(t)\|^2}{\|\gamma'(t)\|^2} - 1 \right) = 0 \quad \square \end{aligned}$$

The function $F(t, x, y)$ is quadratic and $\frac{\partial F}{\partial t}(t, x, y)$ is linear in x and y . Thus, for a fixed $t \in I$, the functions F and $\frac{\partial F}{\partial t}$ can have at most two common zeros according to Bézout's theorem [RG11, p.22 f.]. In the previous proposition, two distinct vanishing points of the family of circles were specified, namely $P_+(t)$ and $P_-(t)$. Hence, these are the only possible zeros of $F(t, x, y)$ that are part of the discriminant envelope.

Theorem 2.4.2 (Equality of E_ℓ and E_d) [BG92, 5.7(6)]

For a family of circles \mathcal{F} as defined in Equation (2.6), the discriminant envelope E_d as defined in Definition 2.2.2 and the limiting envelope E_ℓ from Equation (2.11) are equal:

$$E_\ell = E_d.$$

As a result, the properties of both sets can be examined equally to gain knowledge about the boundary characteristics of a stroke.

2.4.2 Projection of a manifold

We analyze from the perspective of the discriminant envelope where the boundary has singular points, since this is where our algorithm presented in Chapter 5 runs into special cases. We will identify additional properties of a stroke's boundary compared to previous results from the perspective of the limiting envelope.

To examine singular points on the discriminant envelope, we follow the chapter *Local structure of envelopes* in the book by Bruce and Giblin [BG92].

We have previously stated in Equation (2.8) that for all circles along the curve γ , the value zero is regular for $F(t, x, y)$, i.e., $\frac{\partial F}{\partial x}$ and $\frac{\partial F}{\partial y}$ are not simultaneously zero. Therefore, without loss of generality, we assume regularity at zero for $F(t, x, y)$ in further investigations.

First, we study the manifold of preimages $(t, x, y) \in \mathbb{R}^3$ of $F(t, x, y) = 0$ and denote it by $M = F^{-1}(0)$. Figure 2.10 on the left shows M for the Benchmark Example 2.1.6. In $\mathbb{R} \times \mathbb{R}^2$, M is a parameterized 2-manifold. This means that locally around each point (t_0, x_0, y_0) in M , there exists a homeomorphism φ of an open subset $U \subset \mathbb{R}^2$ to M [VGdF07, Chapter 2]. This map φ is an immersion, i.e., its derivative has full rank. Thus, M arises locally from an open set $U \subset \mathbb{R}^2$ by this map φ . Specifically, for the map $\varphi: \mathbb{R}^2 \rightarrow \mathbb{R} \times \mathbb{R}^2$, $U \ni (x, y) \mapsto (t, x, y) \in M$, the derivative of φ has rank two and $F(\varphi(x, y)) = 0$.

Projecting $M \subseteq \mathbb{R} \times \mathbb{R}^2$ onto \mathbb{R}^2 by $\pi: \mathbb{R} \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $(t, x, y) \mapsto (x, y)$, results in the collection of circles that define the corresponding stroke. Figure 2.10 on the right shows a visualization of $\pi(M)$ for Benchmark Example 2.1.6.

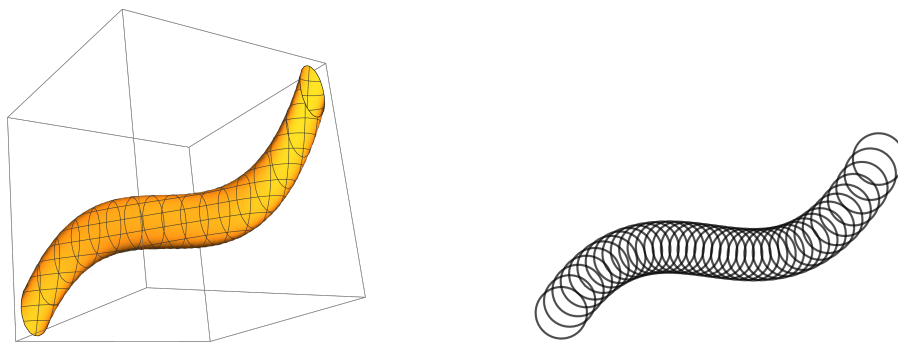


Figure 2.10: Manifold $M = F^{-1}(0)$ and its projection $\pi(M) \in \mathbb{R}^2$.

The discriminant envelope of a stroke is a subset of the projection of M onto \mathbb{R}^2 . To investigate the characteristics of the envelopes, we analyze the map $\pi \circ \varphi: U \rightarrow \pi(M)$. The composition $\pi \circ \varphi$ is a diffeomorphism, i.e., it is continuously differentiable and invertible with a continuously differentiable inverse if and only if its derivative with respect to t is non-zero [BG92, Proposition 5.20].

Proposition 2.4.3 *Given an open subset U of the real plane \mathbb{R}^2 , a parameterized 2-manifold $M \subseteq \mathbb{R} \times \mathbb{R}^2$, the map $\varphi: U \rightarrow M$ and the projection $\pi: M \rightarrow \mathbb{R}^2$, $(t, x, y) \mapsto (x, y)$. Then $\pi \circ \varphi$ is a local diffeomorphism at point $(x_0, y_0) \in U$ if and only if $\frac{\partial F}{\partial t}(t_0, x_0, y_0) \neq 0$.*

Proof: The Jacobian of the projection π is $J_\pi = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. The Jacobian A at (x_0, y_0) of the composition $\pi \circ \varphi$ corresponds to the last two columns of the Jacobian of φ at (x_0, y_0) by the chain rule:

$$A(x_0, y_0) = J_\pi J_\varphi(x_0, y_0) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} = \begin{pmatrix} c & d \\ e & f \end{pmatrix}.$$

Hence, we need to show that A is invertible if and only if $\frac{\partial F}{\partial t}(t_0, x_0, y_0) \neq 0$, which in turn is equivalent to the condition that $(1, 0, 0)$ is not in the kernel of the differential of $F(t_0, x_0, y_0)$. Since $F(\varphi(x_0, y_0)) = 0$, the kernel of $DF(t_0, x_0, y_0)$ corresponds to the image of $D\varphi(x_0, y_0)$. There is a solution of

$$J_\varphi(x_0, y_0) \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} a & b \\ A \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

if and only if A is singular based on arguments of linear independence and the fact that $D\varphi$ has rank two. This completes the proof since this implies that $(1, 0, 0)$ is not in the image of $D\varphi(x_0, y_0)$ if and only if A is invertible. \square

This proposition shows that points (x, y) in E_d never originate from preimage points in U under the diffeomorphism $\pi \circ \varphi$ as in Proposition 2.4.3, since for $(x, y) \in E_d$ it must hold that $F(t, x, y) = \frac{\partial F}{\partial t}(t, x, y) = 0$.

Following Bruce and Giblin [BG92, 5.21], we call $(t, x, y) \in M$ a *critical point* of $\pi: M \rightarrow \mathbb{R}^2$ if the composition $\pi \circ \varphi$ is not a diffeomorphism for the preimage of (t, x, y) in U . A *critical value* of the projection π is a point $(x, y) \in \mathbb{R}^2$ for which its preimage $\pi^{-1}(x, y) = (t, x, y)$ is a critical point of π .

We define Σ as the set of all critical points (t, x, y) of π in M . This set $\Sigma \subset \mathbb{R} \times \mathbb{R}^2$ is composed of points for which $F(t, x, y) = \frac{\partial F}{\partial t}(t, x, y) = 0$. The discriminant envelope E_d is the projection of Σ onto the real plane \mathbb{R}^2 via π :

$$E_d = \pi(\Sigma).$$

For the manifold M shown in Figure 2.10, the critical points consist of those (t, x, y) where M intersects the manifold $\partial M = \left(\frac{\partial F}{\partial t}\right)^{-1}(0)$ which is visualized in Figure 2.11. The projection of these points with π gives the expected envelope of the family of circles.

Hence, the behavior of a stroke's envelope can also be studied by looking at the manifold M and the set Σ . For this, following [BG92, 5.21], let G be the map containing information about whether a point $(t, x, y) \in M$ is critical and hence in Σ or not:

$$G: \mathbb{R} \times \mathbb{R}^2 \rightarrow \mathbb{R}^2, \quad (t, x, y) \mapsto \left(F(t, x, y), \frac{\partial F}{\partial t}(t, x, y) \right).$$

It holds that Σ is the set of all preimage points of $(0, 0)$ with respect to G :

$$\Sigma = G^{-1}(0, 0).$$

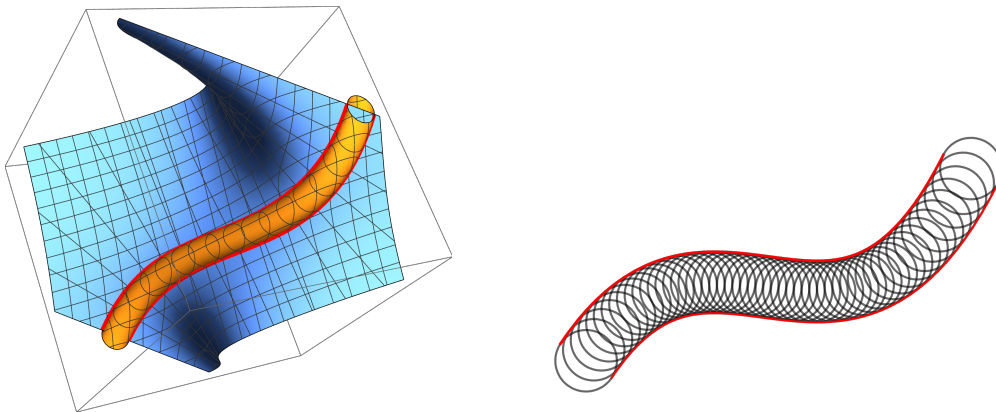


Figure 2.11: Orange manifold $M = F^{-1}(0)$ intersects blue $\partial M = \left(\frac{\partial F}{\partial t}\right)^{-1}(0)$ in red Σ .

Therefore, we can explore the characteristics of G to learn more about Σ . So we take the derivative of G and get its Jacobian

$$J_G = \begin{pmatrix} \frac{\partial F}{\partial t} & \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} \\ \frac{\partial^2 F}{\partial t^2} & \frac{\partial^2 F}{\partial x \partial t} & \frac{\partial^2 F}{\partial y \partial t} \end{pmatrix}$$

at some point (t_0, x_0, y_0) . If $(t_0, x_0, y_0) \in \Sigma$, the partial derivative of F w.r.t. t is zero. Equation (2.8) shows that at least one partial derivative of F w.r.t. x and y is not equal to zero. Hence, the Jacobian of G has full row rank if $\frac{\partial^2 F}{\partial t^2}(t, x, y) \neq 0$, and it follows that $\Sigma = G^{-1}(0, 0)$ has a non-zero derivative at points where the second partial derivative of F with respect to t is non-zero [BG92, 5.22, 4.12, and 4.16]. This implies that Σ is locally a smooth curve for $\frac{\partial^2 F}{\partial t^2}(t, x, y) \neq 0$.

Lemma 2.4.4 *If $\frac{\partial^2 F}{\partial t^2}(t, x, y) \neq 0$, then Σ is locally a parameterized 1-manifold.*

To determine if the smoothness result also applies to the projection $E_d = \pi(\Sigma)$, we examine the local structure of Σ more closely.

2.4.3 Local structure of E_d

We focus on analyzing the local structure of $\Sigma \subseteq M$ and its projection $\pi(\Sigma) = E_d$ onto the discriminant envelope of the family of curves \mathcal{F} in the xy -plane. Globally, several values of t may correspond to the same point (x, y) , which has no effect on the smoothness of Σ in space. But when Σ is projected onto the xy -plane, the point (x, y) becomes a multiple point of the discriminant envelope. If \mathcal{F} is the stroke-defining family of circles, this is equivalent to a self-intersecting stroke. Self-intersecting strokes will be discussed in Sections 4.3.1 and 5.4.1. Until then, we examine the properties of local smoothness and local singularities of Σ and the corresponding discriminant envelope $E_d = \pi(\Sigma)$. For this, we follow Bruce and Giblin [BG92, Proposition 5.25].

Let $(t_0, x_0, y_0) \in \Sigma = G^{-1}(0, 0)$ be a point where the second partial derivative of curve F w.r.t. t is non-zero, i.e., $\frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0) \neq 0$. By definition of Σ , both $F(t_0, x_0, y_0)$ and $\frac{\partial F}{\partial t}(t_0, x_0, y_0)$ vanish. Around point (t_0, x_0, y_0) , there is a neighborhood $V \subset \mathbb{R} \times \mathbb{R}^2$ such that the projection of this neighborhood intersected with Σ is a parameterized 1-manifold in \mathbb{R}^2 , i.e., it is a smooth curve.

To show this, we recall that by Equation (2.8) either $\frac{\partial F}{\partial x}$ or $\frac{\partial F}{\partial y}$ is non-zero at (t_0, x_0, y_0) . For more clarity in the argumentation, we temporarily change the order of variables to (x_0, y_0, t_0) . Without loss of generality, let $\frac{\partial F}{\partial y}(x_0, y_0, t_0) \neq 0$. Then the Jacobian of G evaluated at (x_0, y_0, t_0) is given by

$$J_G(x_0, y_0, t_0) = \begin{pmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} & \frac{\partial F}{\partial t} \\ \frac{\partial^2 F}{\partial x \partial t} & \frac{\partial^2 F}{\partial y \partial t} & \frac{\partial^2 F}{\partial t^2} \end{pmatrix} (x_0, y_0, t_0) = \begin{pmatrix} \bullet & \star & 0 \\ \bullet & \bullet & \star \end{pmatrix} \quad (2.19)$$

where $\bullet \in \mathbb{R}$ and $\star \in \mathbb{R} \setminus \{0\}$. Thus, $J_G(x_0, y_0, t_0)$ has full row rank, and G is a submersion, i.e., it has a surjective differential.

Thus, the Implicit Function Theorem can be applied. Further information about the theorem can, for example, be found in [KP02]. It states that given a continuously differentiable function $f: \mathbb{R}^{n+m} \rightarrow \mathbb{R}^m$ where $(x, y) = (x_1, \dots, x_n, y_1, \dots, y_m) \in \mathbb{R}^{n+m}$, then there exists a continuously differentiable function $g: \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that $f(x, g(x)) = 0$. To construct g , assume that $f(a, b) = 0$. If the Jacobian of f restricted to the derivatives with respect to y

$$J_f|_y(a, b) = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_m} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial y_1} & \cdots & \frac{\partial f_m}{\partial y_m} \end{pmatrix} (a, b)$$

is invertible, then there exists an open neighborhood $V \subset \mathbb{R}^n$ with $a \in V$ such that $g: V \rightarrow \mathbb{R}^m$ with $f(x, g(x)) = 0$ is unique.

Applying this result to our $G: (x, y, t) \rightarrow \mathbb{R}^2$ requires $a = x_0$ and $b = (y_0, t_0)$ with $G(a, b) = G(x_0, y_0, t_0) = 0$. The Jacobian restricted to y and t is given by Equation (2.19):

$$J_G|_{y,t}(b, a) = \begin{pmatrix} \frac{\partial F}{\partial y} & \frac{\partial F}{\partial t} \\ \frac{\partial^2 F}{\partial y \partial t} & \frac{\partial^2 F}{\partial t^2} \end{pmatrix} (x_0, y_0, t_0) = \begin{pmatrix} \star & 0 \\ \bullet & \star \end{pmatrix}.$$

Since this matrix is invertible, there is a neighborhood V of (x_0, y_0, t_0) such that on $\Sigma \cap V$, both y and t can be uniquely written in terms of x using the Implicit Function Theorem. Since y can be expressed as the result of a continuously differentiable function in x , we can define a local diffeomorphism $\phi: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^2$ that maps (x, \tilde{y}) to $(x, \tilde{y} + g(x))$. Hence, the diffeomorphism ϕ maps $(x, 0)$ to $(x, g(x)) = (x, y)$, and the projection $\pi(\Sigma \cap V)$ is locally homeomorphic to \mathbb{R} , so it is a parameterized 1-manifold. We will now return to the ordering of variables (t, x, y) .

Proposition 2.4.5 (Regularity of the envelope)

For $(t_0, x_0, y_0) \in \Sigma$, the projection $\pi(\Sigma \cap V)$ of a neighborhood V of (t_0, x_0, y_0) yields a regular part of the discriminant envelope if $\frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0) \neq 0$.

This means that the discriminant envelope of a stroke is regular for points with $F = \frac{\partial F}{\partial t} = 0$ and $\frac{\partial^2 F}{\partial t^2} \neq 0$. The points where the second derivative of F with respect to t becomes zero define the non-regular points of the discriminant envelope. They are called *points of regression*.

Definition 2.4.6 (Points of regression) [BG92, Definition 5.26]

The points $(x_0, y_0) \in E_d$ for which there is a parameter $t_0 \in I$ with $(t_0, x_0, y_0) \in \Sigma$ and $\frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0) = 0$ are called *points of regression*. The remaining points within the discriminant envelope, where $\frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0) \neq 0$, are called *regular*.

For a neighborhood of a regular point (x_0, y_0) of the discriminant envelope, i.e., for a regular part of $\pi(\Sigma \cap V)$ with V a neighborhood of (t_0, x_0, y_0) , we show that the tangent space of $\pi(\Sigma \cap V)$ at (x_0, y_0) is the same as the tangent space at (x_0, y_0) on the circle $\mathcal{C}(t_0)$ defined by $F(t_0, x, y) = 0$. For this, we follow [BG92, Proposition 5.25]. Note that the tangent space of $\pi(\Sigma \cap V)$ in (x_0, y_0) is the same as the projected tangent space of $\Sigma \cap V$ in (t_0, x_0, y_0) . The tangent space of $\Sigma \cap V$ in (t_0, x_0, y_0) is characterized as follows. We recall that $\Sigma = G^{-1}(0, 0)$, which is why Σ is a level surface of G . It is a known fact from calculus that the gradient of G at a point $(t_0, x_0, y_0) \in \Sigma$ is perpendicular to the tangent space TG of the level surface in (t_0, x_0, y_0) [Kön13, Chapter 2.1, V.]. The Jacobian J_G of G is the transpose of its gradient ∇G . Thus, we can write

$$\nabla G(t_0, x_0, y_0)^T v = J_G(t_0, x_0, y_0) \cdot v = (0, 0)^T$$

for all tangent vectors $v \in TG(t_0, x_0, y_0)$. This implies that any vector $v = (v_1, v_2, v_3)^T$ that is tangent to Σ at (t_0, x_0, y_0) is contained in the kernel of the Jacobian of G at (t_0, x_0, y_0) . With Equation (2.19), it holds

$$\begin{aligned} \overbrace{\frac{\partial F}{\partial t}(t_0, x_0, y_0)}^{=0} v_1 + \frac{\partial F}{\partial x}(t_0, x_0, y_0) v_2 + \frac{\partial F}{\partial y}(t_0, x_0, y_0) v_3 &= 0 \\ \frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0) v_1 + \frac{\partial^2 F}{\partial x \partial t}(t_0, x_0, y_0) v_2 + \frac{\partial^2 F}{\partial y \partial t}(t_0, x_0, y_0) v_3 &= 0. \end{aligned}$$

The first partial derivative of F , $\frac{\partial F}{\partial t}(t_0, x_0, y_0)$, equals zero by definition of Σ . Therefore, the first equation provides a condition on v_2 and v_3 only. The second partial derivative of F , $\frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0)$, is different from zero by assumption. Hence, the second equation gives a condition for v_1 in terms of v_2 and v_3 . Thus, if the tangent space TG at (t_0, x_0, y_0) is projected to the xy -plane, i.e., $(v_1, v_2, v_3)^T \xrightarrow{\pi} (v_2, v_3)$, then the first equation determines which vectors (v_2, v_3) are contained in the tangent space of $\pi(\Sigma \cap V)$ at (x_0, y_0) . More precisely, it contains exactly those (v_2, v_3) with $\frac{\partial F}{\partial x} v_2 + \frac{\partial F}{\partial y} v_3 = 0$. The second equation is no longer relevant because for any (v_2, v_3) we could compute a v_1 such that the second equation holds. Finally, this shows that the vectors (v_2, v_3) in the tangent space of $\pi(\Sigma \cap V)$ at (x_0, y_0) are exactly those in the tangent space of $\mathcal{C}(t_0)$ at (x_0, y_0) , since by the same argument from calculus these are the vectors (v_2, v_3) with $\left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}\right) \begin{pmatrix} v_2 \\ v_3 \end{pmatrix} = 0$.

Hence, for regular points in $\pi(\Sigma) = E_d$, the discriminant envelope coincides with the tangent envelope E_t .

Proposition 2.4.7 (Equality of E_d and E_t for regular envelope points)

If (t_0, x_0, y_0) is a regular point of the discriminant envelope, i.e., $\frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0) \neq 0$, then the discriminant envelope at (x_0, y_0) is tangent to the circle $\mathcal{C}(t_0)$. Hence, for regular parts of the envelope, the discriminant envelope E_d coincides with the tangential envelope E_t .

This implies that by Theorem 2.4.2 and Proposition 2.4.7, we know for regular points of the envelopes of strokes that all three definitions of envelopes coincide:

$$E_\ell = E_d = E_t. \tag{2.20}$$

2.5 Criteria for singularities of the envelopes

For the further analysis of the envelopes of strokes, we again study the points of regressions, i.e., the non-regular envelope points in detail. If a stroke has a constant radius function $r \equiv r_c$, we already know from Theorem 2.3.3 that the points of regression are precisely those where the envelope curve coincides with the evolute of $\gamma(t)$. The points of regression calculated in Bruce and Giblin [BG92, Section 5.30] based on the discriminant envelope E_d are consistent with our results from the perspective of the limiting envelope E_ℓ . Since we have shown in Theorem 2.4.2 that both envelopes coincide for our strokes, the equality of the sets of points of regression is a confirmation.

For a non-constant radius function, we found in Proposition 2.3.5 that the limiting envelope E_ℓ for an arc-length parameterized curve γ has singular points where the curvature of γ satisfies a certain criterion. In view of the fact that $E_\ell = E_d$ has been shown, we want to derive the same from the perspective of the discriminant envelope and, additionally, study the singular points of non arc-length parameterized curves γ .

The family of circles \mathcal{F} defining a stroke is given by Equation (2.6) as

$$F(t, x, y) = \|(x, y)^T - \gamma(t)\|^2 - r(t)^2$$

for curve γ , radius function r and parameter $t \in I$.

From Theorem 2.4.2, it is known that the points in E_ℓ , as defined in Equation (2.11), are exactly those in the set E_d , i.e., they satisfy the conditions $F = \frac{\partial F}{\partial t} = 0$. Hence, to identify the points of regression in the discriminant envelope, we search for those points $(x_\ell, y_\ell) \in E_\ell$ that satisfy $\frac{\partial^2 F}{\partial t^2}(t, x_\ell, y_\ell) = 0$ for some $t \in I$. From the proof of Proposition 2.4.1, we know that

$$\frac{\partial F}{\partial t}(t, x, y) = -2 \langle (x, y)^T - \gamma(t), \gamma'(t) \rangle - 2r(t)r'(t)$$

and it follows

$$\frac{\partial^2 F}{\partial t^2}(t, x, y) = 2\|\gamma'(t)\|^2 - 2 \langle (x, y)^T - \gamma(t), \gamma''(t) \rangle - 2r'(t)^2 - 2r(t)r''(t).$$

We search for a point

$$(x_\ell, y_\ell) = \gamma(t) - \frac{r(t)r'(t)}{\|\gamma'(t)\|^2} \gamma'(t) \pm \frac{r(t)}{\|\gamma'(t)\|^2} \sqrt{\|\gamma'(t)\|^2 - r'(t)^2} (\gamma'(t))^\perp$$

for some $t \in I$ where the second derivative of F with respect to t equals zero.

To simplify calculations, we temporarily assume that γ is arc-length parameterized:

$$\|\gamma'(t)\| = \sqrt{\gamma'_x(t)^2 + \gamma'_y(t)^2} = 1.$$

This reduces $\frac{\partial^2 F}{\partial t^2}$ and (x_ℓ, y_ℓ) to

$$\begin{aligned} \frac{\partial^2 F}{\partial t^2}(t, x, y) &= 2(1 - r'(t)^2 - r(t)r''(t) - \langle (x, y)^T - \gamma(t), \gamma''(t) \rangle) \\ (x_\ell, y_\ell) &= \gamma(t) - r(t)r'(t)\gamma'(t) \pm r(t)\sqrt{1 - r'(t)^2} (\gamma'(t))^\perp. \end{aligned} \tag{2.21}$$

Plugging those points (x_ℓ, y_ℓ) into the second derivative of F w.r.t. t yields

$$0 \stackrel{!}{=} \frac{\partial^2 F}{\partial t^2}(t, x_\ell, y_\ell) = 2 \cdot \left[1 - r'(t)^2 - r(t)r''(t) - \left(-r(t)r'(t) \langle \gamma'(t), \gamma''(t) \rangle \pm r(t) \sqrt{1 - r'(t)^2} \langle \gamma'(t)^\perp, \gamma''(t) \rangle \right) \right].$$

This equation can be simplified using two principles from differential geometry for planar curves. The curvature of γ is defined by

$$\kappa(t) = \frac{\det(\gamma'(t), \gamma''(t))}{\|\gamma'(t)\|^3} \stackrel{\|\gamma'(t)\|=1}{=} \det(\gamma'(t), \gamma''(t)) = \langle \gamma'(t)^\perp, \gamma''(t) \rangle. \quad (2.22)$$

Furthermore, since γ is arc-length parameterized, it holds $\langle \gamma'(t), \gamma'(t) \rangle = \|\gamma'(t)\|^2 = 1$, and hence

$$\frac{1}{2} \frac{\partial}{\partial t} \langle \gamma'(t), \gamma'(t) \rangle = \langle \gamma'(t), \gamma''(t) \rangle = 0.$$

This gives

$$\begin{aligned} 0 &= \frac{\partial^2 F}{\partial t^2}(t, x_\ell, y_\ell) = 2 \left(1 - r'(t)^2 - r(t)r''(t) \mp r(t) \sqrt{1 - r'(t)^2} \kappa(t) \right) \\ \Leftrightarrow \kappa(t) &= \pm \frac{1 - r'(t)^2 - r(t)r''(t)}{r(t) \sqrt{1 - r'(t)^2}} \end{aligned}$$

We are not dividing by zero because we restricted the radius function to be greater than zero in Section 2.1, and by Proposition 2.1.3, we know that $r'(t)^2 < \|\gamma'(t)\|^2 = 1$. In summary, the above argumentation leads to the following result. This result corresponds to Proposition 2.3.5 for the singular points in the limiting envelope of a stroke.

Proposition 2.5.1 (Points of regression for arc-length parameterized curve)

Let curve γ be arc-length parameterized, and let the radius function r satisfy the Condition (2.2): $r'(t)^2 < \|\gamma'(t)\|^2$. Then the points of regression of the envelope of a family of circles \mathcal{F} defined by $F(t, x, y) = \|(x, y)^T - \gamma(t)\|^2 - r(t)^2 = 0$ are exactly those points $(x, y) \in E_d = E_\ell$ for $t \in I$ for which the curvature of γ satisfies the equation

$$\kappa(t) = \pm \frac{1 - r'(t)^2 - r(t)r''(t)}{r(t) \sqrt{1 - r'(t)^2}}. \quad (2.23)$$

If the curve γ is not arc-length parameterized, the same calculations lead to a similar condition on the curvature of γ that must be satisfied for points of regression:

Proposition 2.5.2 (Points of regression for general curves)

Given a family of circles \mathcal{F} defined by $F(t, x, y)$ and a radius function r as in Proposition 2.5.1, and given a curve γ that is not necessarily arc-length parameterized. Then the points of regression of the envelope of \mathcal{F} are defined as those $(x, y) \in E_\ell = E_d$ for $t \in I$ for which it holds

$$0 = 2 \left(\|\gamma'(t)\|^2 - r'(t)^2 - r(t)r''(t) \mp \|\gamma'(t)\| \cdot r(t) \sqrt{\|\gamma'(t)\|^2 - r'(t)^2} \cdot \kappa(t) + \frac{r(t)r'(t)}{\|\gamma'(t)\|^2} \langle \gamma'(t), \gamma''(t) \rangle \right)$$

$$\Leftrightarrow \kappa(t) = \pm \frac{\|\gamma'(t)\|^2 - r'(t)^2 - r(t)r''(t) + \frac{r(t)r'(t)}{\|\gamma'(t)\|} \langle \gamma'(t), \gamma''(t) \rangle}{\|\gamma'(t)\| \cdot r(t) \sqrt{\|\gamma'(t)\|^2 - r'(t)^2}}$$

As in Chapters 2.3.1 and 2.3.2, these criteria relate points of regression to the curvature of the curve γ . Equipped with the knowledge of the position of the point of regression, we can now analyze the type of singularity that occurs at these points of regression for both constant and non-constant radius functions.

2.6 Type of points of regression in the envelopes

To examine the analytic behavior of the points of regression in the envelopes of our families of circles and thus of our strokes, we consider unfoldings.

Unfoldings exist for functions that depend on different parameters. As an introductory example, consider a quadratic function $f: \mathbb{R} \rightarrow \mathbb{R}$ with $f(t) = t^2 + c_0$ for some constant $c_0 \in \mathbb{R}$. Then, the function $\mathbb{F}: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, $\mathbb{F}(t, c) = t^2 + c$ for $c \in \mathbb{R}$ is a 1-parameter unfolding of f . This unfolding \mathbb{F} includes all unit parabolas translated along the y -axis, and function f is one specific function of this set. The general definition of an unfolding is as follows:

Definition 2.6.1 (n -parameter unfolding) [BG92, p.116]

Let $\mathbb{F}: \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ with $(t, x) \in \mathbb{R} \times \mathbb{R}^n$ denote a smooth function. For a fixed $x_0 \in \mathbb{R}^n$, we define $f(t) := \mathbb{F}(t, x_0): \mathbb{R} \rightarrow \mathbb{R}$. Then \mathbb{F} is called an n -parameter unfolding of f .

For us, it is interesting to work with unfoldings because there is a theory associated with unfoldings that deals with singularities. We will apply this theory to categorize the points of regression of the envelopes of our strokes in Sections 2.6.1 and 2.6.2. We will introduce the necessary results under the assumption that the envelopes of our strokes are parallels $\gamma_{\parallel \pm r_c}$ to an arc-length parameterized curve γ , i.e., $\|\gamma'(t)\| = 1$, with constant radius $r(t) \equiv r_c \in \mathbb{R}_{>0}$. The results will be generalized to strokes with non-constant radius function r in the second parts of 2.6.1 and 2.6.2.

Based on the research of Bruce and Giblin [BG92, 7.12], we examine a stroke defined by the function

$$F(t, x, y) = \|(x, y)^T - \gamma(t)\|^2 - r_c^2 = \langle (x, y)^T - \gamma(t), (x, y)^T - \gamma(t) \rangle - r_c^2 = 0 \quad (2.24)$$

for an arc-length parameterized curve $\gamma(t): I \rightarrow \mathbb{R}^2$ and constant radius $r_c \in \mathbb{R}_{>0}$. The envelope of this stroke consists of the two parallels $\gamma_{\parallel \pm r_c}(t)$ of γ at a distance of $\pm r_c$. These parallels have a singular point if and only if this point also lies on the evolute of γ according to Theorem 2.3.3, i.e., the envelopes have a point of regression at (t_0, x_0, y_0) if and only if

$$\pm r_c = \frac{1}{\kappa(t_0)} \quad \text{and} \quad (x_0, y_0) = \gamma(t_0) + \frac{N(t_0)}{\kappa(t_0)}. \quad (2.25)$$

Furthermore, as per Definition 2.4.6, when (t_0, x_0, y_0) denotes a point of regression, F and its first and second derivatives with respect to t are zero:

$$F(t_0, x_0, y_0) = \frac{\partial F}{\partial t}(t_0, x_0, y_0) = \frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0) = 0.$$

We will show that $F(t) := F(t, x_0, y_0)$ for fixed (x_0, y_0) and variable parameter t is comparable to the basic function $g(t) = t^3$ and that their unfoldings have equal discriminant envelopes up to a diffeomorphism. Hence, we investigate why these discriminant envelopes are the same and what this fact implies for the type of singularities on the boundaries of our strokes.

To define the type of singularity of an unfolding of a general smooth function f , we use the concept of \mathcal{R} -equivalence.

Definition 2.6.2 (\mathcal{R} -equivalence) [BG92, 3.1]

Let $f: \mathbb{R} \rightarrow \mathbb{R}$ be a smooth function. Then f is called \mathcal{R} -equivalent at $t_f \in \mathbb{R}$ to another function $g: \mathbb{R} \rightarrow \mathbb{R}$ at $t_g \in \mathbb{R}$ if there are two open intervals $U_f \subset \mathbb{R}$ and $U_g \subset \mathbb{R}$ within the domains of f and g , respectively, such that there is a diffeomorphism $h: U_f \rightarrow U_g$ with $h(t_f) = t_g$ and $f(t) = g(h(t)) + c$ for $t \in U_f$ and some constant $c \in \mathbb{R}$.

The function f is \mathcal{R} -equivalent at t_0 to $g(t) = \pm t^{k+1}$ at 0 for some $k \geq 0$ if and only if $f^{(p)}(t_0) = 0$ for all derivatives of order $1 \leq p \leq k$ and $f^{(k+1)}(t_0) \neq 0$ [BG92, 3.3]. The sign of g is positive if $f^{(k+1)}(t_0) > 0$ and negative if $f^{(k+1)}(t_0) < 0$. This leads to the definition of an A_k singularity of a smooth function f at parameter t_0 .

Definition 2.6.3 (A_k singularity) [BG92, 3.6]

Let $f: \mathbb{R} \rightarrow \mathbb{R}$ be \mathcal{R} -equivalent at t_0 to $g(t) = \pm t^{k+1}$ at 0 for some $k \geq 0$. Then f has an A_k singularity at t_0 . It is also said that f has type A_k . Furthermore, f has type $A_{\geq k}$ if $f^{(p)}(t_0) = 0$ for all $1 \leq p \leq k$ and f has an A_ℓ singularity for some $\ell \geq k$.

With this definition, we can determine two cases of points of regression of the envelopes of our strokes. In the first case, $F(t) = F(t, x_0, y_0)$, its derivative and also its second derivative w.r.t. t vanish, i.e., $F' = \frac{\partial F}{\partial t} = \frac{\partial^2 F}{\partial t^2} = 0$, but the third derivative is non-zero. This is the case of an A_2 singularity. In the second case, also the third and probably also higher derivatives of F w.r.t. t are zero. Then F has an $A_{>3}$ singularity. In the next two sections, we will examine the properties of these two types of singularities. We will see that the envelope of a stroke has a cusp if F has an A_2 singularity. If F has type $A_{\geq 3}$, the singular point has infinite curvature.

2.6.1 A_2 singularities of discriminant envelopes

Let the function $F(t) = F(t, x_0, y_0)$ have an A_2 singularity at t_0 for fixed (x_0, y_0) , where (t_0, x_0, y_0) is a point of regression. To establish a criterion on F for A_2 singularity, we investigate when the third derivative of F is non-zero:

$$\frac{\partial^3 F}{\partial t^3}(t_0, x_0, y_0) \neq 0.$$

Since γ is arc-length parameterized and the radius function is constant, we know that $T(t) = \gamma'(t)$, $T'(t) = \gamma''(t) = \kappa(t)N(t)$, and we deduce from Equation (2.24) that

$$\begin{aligned} \frac{\partial^2 F}{\partial t^2}(t, x, y) &= -2 \cdot \left\langle \begin{pmatrix} x \\ y \end{pmatrix} - \gamma(t), \gamma''(t) \right\rangle \\ \Leftrightarrow \frac{1}{2} \frac{\partial^2 F}{\partial t^2}(t, x, y) &= - \left\langle \begin{pmatrix} x \\ y \end{pmatrix} - \gamma(t), \kappa(t)N(t) \right\rangle. \end{aligned}$$

Hence, the third derivative of F w.r.t. t at (t_0, x_0, y_0) is given by

$$\frac{1}{2} \frac{\partial^3 F}{\partial t^3}(t_0, x_0, y_0) = \kappa(t_0) \left\langle \overbrace{\gamma'(t_0)}^{T(t_0)}, N(t_0) \right\rangle - \left\langle \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \gamma(t_0), \kappa'(t_0)N(t_0) + \kappa(t_0)N'(t_0) \right\rangle.$$

The first summand is zero due to the perpendicularity of the tangent and normal vector: $\langle T(t), N(t) \rangle = 0$. Furthermore, (t_0, x_0, y_0) is a point of regression and thus it holds that $(x_0, y_0)^T = \gamma(t_0) + \frac{N(t_0)}{\kappa(t_0)}$ as well as $\kappa(t_0) \neq 0$ by Equation (2.25). Also, $N(t_0)$ has unit length, i.e., $\langle N(t_0), N(t_0) \rangle = 1$, and it follows from the derivative of this scalar product that $\langle N(t_0), N'(t_0) \rangle = 0$. Hence, the second summand reduces to

$$\frac{\kappa'(t_0)}{\kappa(t_0)} \langle N(t_0), N(t_0) \rangle + \langle N(t_0), N'(t_0) \rangle = \frac{\kappa'(t_0)}{\kappa(t_0)} + 0.$$

In total, it holds that the third derivative of F w.r.t. t is non-zero if and only if the curvature κ has a non-zero derivative:

$$\frac{\partial^3 F}{\partial t^3}(t_0, x_0, y_0) \neq 0 \Leftrightarrow \frac{\kappa'(t_0)}{\kappa(t_0)} \neq 0 \Leftrightarrow \kappa'(t_0) \neq 0. \quad (2.26)$$

According to Definition 2.6.3, the A_2 singularity implies that F is diffeomorphic to $g(t) = \pm t^3$ if and only if $\kappa'(t_0) \neq 0$ for the curvature κ of the curve γ at a point of regression (t_0, x_0, y_0) . Also, the unfoldings of F and g are related and behave similarly, which will be examined in the following.

According to Bruce and Giblin [BG92, p.102 f.], there is more than one unfolding for general smooth functions $f(t)$. A function can be part of several unfoldings, e.g., $f(t) = t^4$ is unfolded by $\mathbb{F}_1(t, x, y) = t^4 + x \cdot t - y$ and by $\mathbb{F}_2(t, x, y) = t^4 + x \cdot t^2 + y \cdot t$. An unfolding consisting of all other functions \mathbb{F} close to f is called *universal*.

A universal unfolding for $g(t) = \pm t^{k+1}$ at $t_0 = 0$ is given by [BG92, 6.6]

$$\begin{aligned} \mathbb{G}: \mathbb{R} \times \mathbb{R}^k &\rightarrow \mathbb{R} \\ \mathbb{G}(t, u) &= \pm t^{k+1} + u_1 + u_2 t + \dots + u_k t^{k-1}. \end{aligned} \quad (2.27)$$

There is no term in t^k due to the so-called *Tschirnhausen transformation*, which uses substitution to bring polynomials of degree m to other polynomials of the same degree and leading term but with otherwise fewer summands and a hopefully more accessible or more suitable form for the respective purpose (see [Ser68, p.346 ff.] for more information). For instance, if a cubic polynomial $t^3 + \lambda t^2 + \mu t + \nu$ with degree 3 is given,

the substitution $t \rightarrow t - \frac{1}{3}\lambda$ yields $t^3 + u_1 + u_2t$, where $\lambda, \mu, \nu, u_1, u_2$ are real coefficients [BG92, 5.36].

Besides the universal unfolding $\mathbb{G}(t, x)$ of $g(t) = \pm t^{k+1}$, there is another unfolding of g that can be induced from a general smooth function $f(t)$ with an A_k singularity at t_0 , i.e., $f(t) = g(h(t)) + c$ locally for $h(t)$ and c as in Definition 2.6.2. Let $\mathbb{F}(t, x): \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ be an unfolding of $f(t)$. Then, \mathbb{F} can be reduced to an unfolding of $g(t) = \pm t^{k+1}$ by [BG92, p.104]

$$\bar{\mathbb{F}}(t, x) = \mathbb{F}(h^{-1}(t), x) - c.$$

Hence, for values of t near $t_0 = 0$ and $x_0 \in \mathbb{R}^n$, it follows that $\bar{\mathbb{F}}(t, x_0) = g(t)$.

The unfolding $\bar{\mathbb{F}}(t, x)$ of g can be written in terms of the universal unfolding $\mathbb{G}(t, x)$ of g by

$$\bar{\mathbb{F}}(t, x) = \mathbb{G}(a(t, x), b(x)) \tag{2.28}$$

for $x \in \mathbb{R}^n$, a smooth function $a(t, x): \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ with $a(t, x_0) = t$ for t close to $t_0 = 0$, and a smooth function $b(x): \mathbb{R}^n \rightarrow \mathbb{R}^k$ with $b(x) := (b_1(x), b_2(x), \dots, b_k(x))$, where it is $b_j(x) = u_j$ in Equation (2.27) and $b_j(x_0) = 0$ for all $j \in \{1, \dots, k\}$ [BG92, 6.3]. Hence, $\bar{\mathbb{F}}(t, x)$ is a universal unfolding for g induced from \mathbb{G} .

Note 2.6.4 *The universal unfoldings we consider here are called versal unfoldings in the book by Bruce and Giblin [BG92]. They distinguish between versal unfoldings and (p)versal unfoldings. The main difference is that (p)versal unfoldings $\mathbb{G}(t, x)$ have no constant term x_1 in Equation (2.27), and hence $\mathbb{G}: \mathbb{R} \times \mathbb{R}^{k-1} \rightarrow \mathbb{R}$. The constant is added when $\bar{\mathbb{F}}(t, x)$ is induced from $\mathbb{G}(t, x)$ by $\bar{\mathbb{F}}(t, x) = \mathbb{G}(a(t, x), b(x)) + c(x)$, where a and b are smooth functions as above, and $c: \mathbb{R}^{k-1} \rightarrow \mathbb{R}$ [BG92, 6.3p]. However, since we are interested in the behavior of our families of circles where F and the derivative of F equal zero, it is not reasonable to add a constant, since the resulting unfoldings would no longer be equal to zero. This is also noted in [BG92, 6.4].*

Given that both $\bar{\mathbb{F}}(t, x) = \mathbb{G}(a(t, x), b(x))$ and $\mathbb{G}(t, u)$ are universal unfoldings of $g(t)$, we deduce that there is not only one universal unfolding for a function. The following matrix criterion gives a necessary and sufficient condition for an unfolding \mathbb{F} of a general smooth function f to be universal.

Theorem 2.6.5 (Matrix criterion for universality) [BG92, 6.10]

Let $f(t)$ have an A_k singularity at t_0 and let $\mathbb{F}(t, x)$ be an n -parameter unfolding of f with $f(t) = \mathbb{F}(t, x_0)$. Then $\mathbb{F}(t, x)$ is a universal unfolding of $f(t)$ at t_0 if and only if the following $k \times n$ -matrix $\mathcal{M}_{\mathbb{F}}(t_0, x_0)$ has rank k :

$$\mathcal{M}_{\mathbb{F}}(t_0, x_0) = \begin{pmatrix} \frac{\partial \mathbb{F}}{\partial x_1}(t_0, x_0) & \frac{\partial \mathbb{F}}{\partial x_2}(t_0, x_0) & \cdots & \frac{\partial \mathbb{F}}{\partial x_n}(t_0, x_0) \\ \frac{\partial^2 \mathbb{F}}{\partial x_1 \partial t}(t_0, x_0) & \frac{\partial^2 \mathbb{F}}{\partial x_2 \partial t}(t_0, x_0) & \cdots & \frac{\partial^2 \mathbb{F}}{\partial x_n \partial t}(t_0, x_0) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^k \mathbb{F}}{\partial x_1 \partial t^{k-1}}(t_0, x_0) & \frac{\partial^k \mathbb{F}}{\partial x_2 \partial t^{k-1}}(t_0, x_0) & \cdots & \frac{\partial^k \mathbb{F}}{\partial x_n \partial t^{k-1}}(t_0, x_0) \end{pmatrix}.$$

Note that this is only possible for $n \geq k$.

Using this knowledge for our strokes, we investigate whether $F(t, x, y)$ from Equation (2.24) is a universal unfolding of $F(t) = F(t, x_0, y_0)$ at t_0 , where (t_0, x_0, y_0) is a point of regression. For this, we set up the matrix

$$\mathcal{M}_F(t_0, x_0, y_0) = \begin{pmatrix} \frac{\partial F}{\partial x}(t_0, x_0, y_0) & \frac{\partial F}{\partial y}(t_0, x_0, y_0) \\ \frac{\partial^2 F}{\partial x \partial t}(t_0, x_0, y_0) & \frac{\partial^2 F}{\partial y \partial t}(t_0, x_0, y_0) \end{pmatrix} = \begin{pmatrix} 2(x_0 - \gamma_x(t_0)) & 2(y_0 - \gamma_y(t_0)) \\ -2\gamma'_x(t_0) & -2\gamma'_y(t_0) \end{pmatrix}.$$

Whether this has full rank can be determined by calculating the determinant:

$$\begin{aligned} \det(\mathcal{M}_F(t_0, x_0, y_0)) &= -4(x_0 - \gamma_x(t_0))\gamma'_y(t_0) + 4(y_0 - \gamma_y(t_0))\gamma'_x(t_0) = \\ &= 4 \cdot \left\langle \begin{pmatrix} x_0 - \gamma_x(t_0) \\ y_0 - \gamma_y(t_0) \end{pmatrix}, \begin{pmatrix} -\gamma'_y(t_0) \\ \gamma'_x(t_0) \end{pmatrix} \right\rangle = \\ &= 4 \cdot \langle x_0 - \gamma(t_0), N(t_0) \rangle \stackrel{(2.25)}{=} 4 \cdot \left\langle \frac{N(t_0)}{\kappa(t_0)}, N(t_0) \right\rangle = 4 \frac{1}{\kappa(t_0)}. \end{aligned}$$

The last equality holds due to the arc-length parameterization of γ . The determinant is non-zero due to the criterion given in Equation (2.25) for t_0 at the points of regression. The non-zero radius r_c is equal to the reciprocal of the curvature of γ at t_0 . Hence, the function F that defines the stroke is indeed a universal unfolding for points of regression of the envelope. This is used to identify the type of singularity at points of regression.

We study the connection between the discriminant envelopes E_d of a family of curves and the results of unfoldings. For $F(t) = F(t, x_0, y_0)$, we previously stated that F has an A_2 singularity at a point of regression (t_0, x_0, y_0) if and only if $\kappa'(t_0) \neq 0$ (see Equation (2.26)). Hence there exists a diffeomorphism $h(t)$ connecting $F(t)$ at t_0 to $g(t) = \pm t^3$ at 0 by the definition of \mathcal{R} -equivalence in Definition 2.6.2. Let G be a universal unfolding of $g(t)$. It follows from Bruce and Giblin [BG92, 6.14], [BG92, 6.13p & 6.14p] and Section 2.4.2 that the discriminant envelopes of F and G are diffeomorphic.

We will present this result for the general functions f , \mathbb{F} , g , and \mathbb{G} shortly. For this, we define a set $M_{\mathbb{F}}$ consisting of all vectors in \mathbb{R}^{n+1} where the unfolding \mathbb{F} equals zero:

$$M_{\mathbb{F}} = \{(t, x) \in \mathbb{R} \times \mathbb{R}^n : \mathbb{F}(t, x) = 0\}.$$

The set $M_{\mathbb{G}}$ is defined analogously.

For our strokes, i.e., for the function F , it holds that $(t_0, x_0, y_0) \in M_F$ for points of regression (t_0, x_0, y_0) . Furthermore, the points of regression (t_0, x_0, y_0) are regular points of F by Equation (2.8). Also, let $(0, u_0, v_0)$ be the points of interest for G with $(0, u_0, v_0) \in M_G$. Those points $(0, u_0, v_0)$ are regular values of G since we defined G to be a universal unfolding of $g(t) = \pm t^3$. Therefore, in order to satisfy the rank criterion of the matrix in Theorem 2.6.5, the derivatives of G with respect to both u_0 and v_0 cannot be zero at the same time. It follows from Section 2.4.2 that M_F and M_G are parameterized 2-manifolds of \mathbb{R}^3 in a neighborhood U of (t_0, x_0, y_0) and V of $(0, u_0, v_0)$, respectively. Furthermore, the projections $\pi(M_F) \in \mathbb{R}^2$ and $\pi(M_G) \in \mathbb{R}^2$ for $\pi: \mathbb{R} \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$ with $\pi(t, x, y) = (x, y)$ have critical values exactly for those $(x, y) \in \mathbb{R}^2$ that are in the

discriminant envelopes of F and G , respectively:

$$E_d^F = \left\{ (x, y) \in \mathbb{R}^2 : \exists t \in \mathbb{R} \text{ with } F(t, x, y) = \frac{\partial F}{\partial t}(t, x, y) = 0 \right\}$$

and E_d^G accordingly.

For general unfoldings \mathbb{F} of a smooth function f and a universal unfolding \mathbb{G} of $g(t) = \pm t^{k+1}$, the discriminant envelopes are defined by

$$E_d^{\mathbb{F}} = \left\{ (x, y) \in \mathbb{R}^2 : \exists t \in \mathbb{R} \text{ with } \mathbb{F}(t, x, y) = \frac{\partial \mathbb{F}}{\partial t}(t, x, y) = 0 \right\}$$

and $E_d^{\mathbb{G}}$ accordingly. The discriminant envelopes $E_d^{\mathbb{F}}$ and $E_d^{\mathbb{G}}$ are locally diffeomorphic to each other [BG92, 6.14].

Theorem 2.6.6 (Uniqueness of the discriminant set of universal unfoldings)

Let $\mathbb{F}(t, x)$ and $\mathbb{G}(t, u)$ be universal unfoldings for $t \in \mathbb{R}$, $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^k$ of \mathcal{R} -equivalent functions $f(t) = \mathbb{F}(t, x_0)$ and $g(t) = \mathbb{G}(t, u_0)$ at t_0 and t_1 , respectively, both having an A_k singularity there. Then, for suitable neighborhoods $U \subset \mathbb{R} \times \mathbb{R}^n$ with $(t_0, x_0) \in U$ and $V \subset \mathbb{R} \times \mathbb{R}^k$ with $(t_1, u_0) \in V$, there is a diffeomorphism $\phi: U \rightarrow V$, $\phi(t, x) = (a(t, x), b(x))$ which satisfies $a(t_0, x_0) = t_1$ and $b(x_0) = u_0$. Furthermore, this diffeomorphism locally projects the parameterized manifolds $M_{\mathbb{F}}$ and $M_{\mathbb{G}}$ to one another:

$$\phi(M_{\mathbb{F}} \cap U) = M_{\mathbb{G}} \cap V.$$

Additionally, the function $b: \pi(U) \rightarrow \pi(V)$ is a diffeomorphism that maps the critical set of the projected manifolds onto each other:

$$b(E_d^{\mathbb{F}} \cap \pi(U)) = E_d^{\mathbb{G}} \cap \pi(V).$$

Hence, the discriminant envelopes $E_d^{\mathbb{F}}$ and $E_d^{\mathbb{G}}$ are locally diffeomorphic around (t_0, x_0) and (t_1, u_0) . So we can speak of the one discriminant envelope of an n -parameter unfolding of a function with A_k singularity up to a diffeomorphism.

Proof: (adaption of the proof in [BG92, 6.14p])

Let f and g be \mathcal{R} -equivalent, i.e., $f(t) = g(h(t))$ for a diffeomorphism $h: \mathbb{R} \rightarrow \mathbb{R}$ with $h(t_0) = t_1$. Furthermore, let $\mathbb{G}_1(t, u) = \mathbb{G}(h(t), u)$. In the following, we identify two diffeomorphisms ϕ_1 and ϕ_2 such that the theorem holds for $\mathbb{F} = \mathbb{G}_1$ with $n = k$ and \mathbb{G} together with ϕ_1 as well as for \mathbb{F} and $\mathbb{G} = \mathbb{G}_1$ together with ϕ_2 . Combining both cases results in the diffeomorphism $\phi = \phi_1 \circ \phi_2: (V_1 \rightarrow V) \circ (U \rightarrow V_1): U \rightarrow V$.

For $\phi_1: V_1 \rightarrow V$, we use $\phi(t, u) = (h(t), u)$, which yields the diffeomorphism $a(t, u) = h(t)$ and the identity diffeomorphism $b(u) = u$.

The other component $\phi_2: U \rightarrow V_1$ is more challenging. It addresses the case in which $t_0 = t_1$ and $f = g$, since we assumed that $f(t) = g(h(t))$ by \mathcal{R} -equivalence. Since \mathbb{G} is the universal unfolding of g , there are smooth functions $a(t, x)$ and $b(x)$ as defined in Equation (2.28) such that $\mathbb{F}(t, x) = \mathbb{G}(a(t, x), b(x))$ with $a(t, x_0) = t$ for t close to t_0 . We need to show that ϕ_2 consists of a and b : $\phi_2(t, x) = (a(t, x), b(x))$.

For this, we calculate the derivative of \mathbb{F} with respect to x_i for all $i \in \{1, \dots, n\}$

$$\frac{\partial \mathbb{F}}{\partial x_i}(t, x) = \frac{\partial \mathbb{G}}{\partial t}(t, u) \frac{\partial a}{\partial x_i}(t, x) + \sum_{\ell=1}^k \frac{\partial \mathbb{G}}{\partial u_\ell}(t, u) \frac{\partial b_\ell}{\partial x_i}(x) \quad (2.29)$$

where $\mathbb{G}(t, u) = \pm a(t, x)^{k+1} + b_1(x) + b_2(x)t + \dots + b_k(x)t^{k-1}$. Since $\mathbb{G}(t, u_0)$ has type A_k at t_0 , it follows that $\frac{\partial \mathbb{G}}{\partial t}(t, u_0) = 0$. Thus, if we set $x = x_0$ and $u = u_0$ on both sides, the first summand on the right in Equation (2.29) vanishes. By taking the derivative $k-1$ times with respect to t on both sides of Equation (2.29) and summing over Equation (2.29) and all of its derivatives, we obtain

$$\begin{aligned} \frac{\partial \mathbb{F}}{\partial x_i}(t_0, x_0) + \left(\sum_{j=1}^{k-1} t^j \frac{\partial^{1+j} \mathbb{F}}{\partial x_i \partial t^j}(t, x_0) \right) (t_0) &= \sum_{\ell=1}^k \frac{\partial \mathbb{G}}{\partial u_\ell}(t_0, u_0) \frac{\partial b_\ell}{\partial x_i}(x_0) \\ &+ \sum_{\ell=1}^k \left(\sum_{j=1}^{k-1} t^j \frac{\partial^{1+j} \mathbb{G}}{\partial u_\ell \partial t^j}(t, u_0) \right) (t_0) \frac{\partial b_\ell}{\partial x_i}(x_0). \end{aligned}$$

This equation can be expressed using the matrices $\mathcal{M}_{\mathbb{F}}(t, x_0)$ for $\mathbb{F}(t, x)$ and $\mathcal{M}_{\mathbb{G}}(t, u_0)$ for $\mathbb{G}(t, u)$ as presented in Theorem 2.6.5:

$$\begin{aligned} \begin{pmatrix} t^0 \\ t^1 \\ \vdots \\ t^{k-1} \end{pmatrix}^T \underbrace{\begin{pmatrix} \frac{\partial \mathbb{F}}{\partial x_1}(t, x_0) & \dots & \frac{\partial \mathbb{F}}{\partial x_n}(t, x_0) \\ \frac{\partial^2 \mathbb{F}}{\partial x_1 \partial t}(t, x_0) & \dots & \frac{\partial^2 \mathbb{F}}{\partial x_n \partial t}(t, x_0) \\ \vdots & & \vdots \\ \frac{\partial^k \mathbb{F}}{\partial x_1 \partial t^{k-1}}(t, x_0) & \dots & \frac{\partial^k \mathbb{F}}{\partial x_n \partial t^{k-1}}(t, x_0) \end{pmatrix}}_{\mathcal{M}_{\mathbb{F}}(t_0, x_0)} (t_0) = \\ = \begin{pmatrix} t^0 \\ t^1 \\ \vdots \\ t^{k-1} \end{pmatrix}^T \underbrace{\begin{pmatrix} \frac{\partial \mathbb{G}}{\partial u_1}(t, u_0) & \dots & \frac{\partial \mathbb{G}}{\partial u_k}(t, u_0) \\ \frac{\partial^2 \mathbb{G}}{\partial u_1 \partial t}(t, u_0) & \dots & \frac{\partial^2 \mathbb{G}}{\partial u_k \partial t}(t, u_0) \\ \vdots & & \vdots \\ \frac{\partial^k \mathbb{G}}{\partial u_1 \partial t^{k-1}}(t, u_0) & \dots & \frac{\partial^k \mathbb{G}}{\partial u_k \partial t^{k-1}}(t, u_0) \end{pmatrix}}_{\mathcal{M}_{\mathbb{G}}(t_0, u_0)} (t_0) \underbrace{\begin{pmatrix} \frac{\partial b_1}{\partial x_1}(x_0) & \dots & \frac{\partial b_1}{\partial x_n}(x_0) \\ \frac{\partial b_2}{\partial x_1}(x_0) & \dots & \frac{\partial b_2}{\partial x_n}(x_0) \\ \vdots & & \vdots \\ \frac{\partial b_k}{\partial x_1}(x_0) & \dots & \frac{\partial b_k}{\partial x_n}(x_0) \end{pmatrix}}_{J_b(x_0)}. \end{aligned}$$

We will apply the Inverse Function Theorem, which states that a continuously differentiable function is locally invertible with a continuously differentiable inverse if the Jacobian has a non-zero determinant (see for example [KP02] as a reference). To apply it to b , we assume $n = k$ (which is sufficient for our case of application since we are considering A_2 singularities for $n = 2$). With this, the matrix criterion of Theorem 2.6.5 indicates that $\mathcal{M}_{\mathbb{F}}(t_0, x_0)$ and $\mathcal{M}_{\mathbb{G}}(t_0, u_0)$ have full rank since they are universal unfoldings. Therefore, the Jacobi matrix of $b(x)$ also has full rank at x_0 . By the Inverse Function Theorem, it can be concluded that b is a diffeomorphism.

We use the same approach again and test the Jacobian J_{ϕ_2} of $\phi_2(t, x)$ to see if ϕ_2 is a local diffeomorphism. It holds that

$$J_{\phi_2}(t_0, x_0) = \begin{pmatrix} \frac{\partial a}{\partial t}(t_0, x_0) & \bullet \\ 0 & J_b(x_0) \end{pmatrix}.$$

We already know that $J_b(x_0)$ has full rank. In addition, $a(t, x_0) = t$ by assumption and hence $\frac{\partial a}{\partial t}(t_0, x_0) = 1$ and J_{ϕ_2} has full rank at (t_0, x_0) . This shows that $\phi_2(t, x) = (a(t, x), b(x))$ is a local diffeomorphism and $\phi = \phi_1 \circ \phi_2$ locally maps $M_{\mathbb{F}} \cap U$ to $M_{\mathbb{G}} \cap V$.

It remains to show that b locally maps the discriminant envelopes of \mathbb{F} and \mathbb{G} onto each other. For this, we apply that, locally, ϕ is a diffeomorphism. It follows that $\frac{\partial a}{\partial t}(t, x) \neq 0$ for all $(t, x) \in U$, since the Jacobian J_{ϕ} has full rank. Furthermore, it holds

$$\begin{aligned} \frac{\partial \mathbb{F}}{\partial t}(t, x) &= \frac{\partial \mathbb{G}}{\partial t}(a(t, x), b(x)) \frac{\partial a}{\partial t}(t, x) \\ \frac{\partial^2 \mathbb{F}}{\partial t^2}(t, x) &= \frac{\partial^2 \mathbb{G}}{\partial t^2}(a(t, x), b(x)) \left(\frac{\partial a}{\partial t}(t, x) \right)^2 + \frac{\partial \mathbb{G}}{\partial t}(a(t, x), b(x)) \frac{\partial^2 a}{\partial t^2}(t, x) \\ \Rightarrow \frac{\partial \mathbb{F}}{\partial t}(t, x) = \frac{\partial^2 \mathbb{F}}{\partial t^2}(t, x) = 0 &\Leftrightarrow \frac{\partial \mathbb{G}}{\partial t}(a(t, x), b(x)) = \frac{\partial^2 \mathbb{G}}{\partial t^2}(a(t, x), b(x)) = 0. \end{aligned}$$

Hence, the image point $b(x)$ is inside the discriminant envelope $E_d^{\mathbb{G}} \cap \pi(V)$ if and only if the preimage point x is inside the intersection set $E_d^{\mathbb{F}} \cap \pi(U)$. This concludes the proof of Theorem 2.6.6. \square

We apply the theorem to our strokes and find that, up to a diffeomorphism, we can consider the points of regression of the envelope in the universal unfolding of $g(t) = \pm t^3$ instead of the critical points in the envelope of a stroke with constant radius r_c and arc-length parameterized curve γ . Function g has the universal unfolding $G(t, x, y) = t^3 + x + yt$. Hence, the discriminant set E_d^G is given by [BG92, 6.17p]

$$\begin{aligned} E_d^G &= \{(x, y) \in \mathbb{R}^2 : \exists t \in \mathbb{R} \text{ with } G(t, x, y) = \frac{\partial G}{\partial t}(t, x, y) = 0\} \\ &= \{(x, y) \in \mathbb{R}^2 : \exists t \in \mathbb{R} \text{ with } t^3 + x + yt = 3t^2 + y = 0\} \\ &= \{(x, y) \in \mathbb{R}^2 : \exists t \in \mathbb{R} \text{ with } y = -3t^2, x = -t^3 - yt = 2t^3\} \\ &= \{(x, y) \in \mathbb{R}^2 : 4y^3 + 27x^2 = 0\}. \end{aligned}$$

The equation $4y^3 + 27x^2 = 0$ has a critical point, i.e., a singularity, at zero, which is the point of regression of G (see Figure 2.12). This singularity is called an *ordinary cusp*.

Hence, up to a diffeomorphism, the envelopes E_d^G and E_d^F for $F(t, x, y)$ as in Equation (2.24) have an ordinary cusp at points of regressions (t_0, x_0, y_0) with A_2 singularity.

Corollary 2.6.7 *Let $\gamma: I \rightarrow \mathbb{R}^2$ be an arc-length parameterized curve, $r_c \in \mathbb{R}_{>0}$ and $F(t, x, y) = \|(x, y)^T - \gamma(t)\|^2 - r_c^2$. Then the discriminant envelope E_d of the family of circles \mathcal{F} of F has an ordinary cusp at a point of regression (t_0, x_0, y_0) with $\frac{\partial^3 F}{\partial t^3}(t_0, x_0, y_0) \neq 0$, i.e., at an A_2 singularity of the stroke's boundary.*

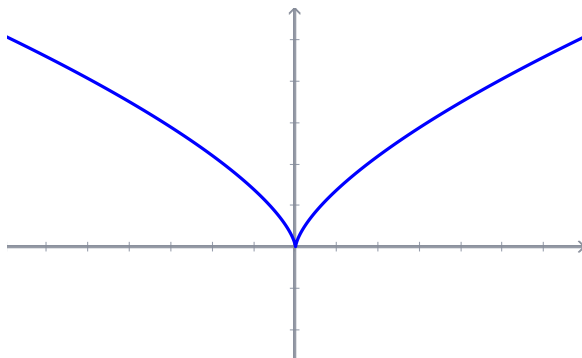


Figure 2.12: The ordinary cusp at zero of $4y^3 + 27x^2 = 0$.

So far, we have only applied the theory of unfoldings to strokes with a constant radius function $r(t) \equiv r_c \in \mathbb{R}_{>0}$. Now, we apply the above results to general radius functions $r: I \rightarrow \mathbb{R}_{>0}$ and non arc-length parameterized curves γ in the definition of $F(t, x, y)$. The points on the envelope of a stroke are given as in Equation (2.11) for limiting envelopes by

$$P_{\pm}(t) = \gamma(t) - \frac{r(t)r'(t)}{\|\gamma'(t)\|^2} \gamma'(t) \pm \frac{r(t)}{\|\gamma'(t)\|^2} \sqrt{\|\gamma'(t)\|^2 - r'(t)^2} (\gamma'(t))^{\perp}.$$

Additionally, the points of regression on the envelope are exactly those where $F(t_0, x_0, y_0) = \frac{\partial F}{\partial t}(t_0, x_0, y_0) = \frac{\partial^2 F}{\partial t^2}(t_0, x_0, y_0) = 0$.

We assume that $\frac{\partial^3 F}{\partial t^3}(t_0, x_0, y_0) \neq 0$ for (t_0, x_0, y_0) , i.e., that F has an A_2 singularity at (t_0, x_0, y_0) . To determine if F is a universal unfolding of $F(t, x_0, y_0)$ and therefore diffeomorphic to the universal unfolding $G(t, x, y) = t^3 + x + yt$ of $g(t) = \pm t^3$, we employ the matrix criterion from Theorem 2.6.5. Thus, we calculate

$$\mathcal{M}_F(t_0, x_0, y_0) = \begin{pmatrix} \frac{\partial F}{\partial x_1}(t_0, x_0, y_0) & \frac{\partial F}{\partial x_2}(t_0, x_0, y_0) \\ \frac{\partial^2 F}{\partial x_1 \partial t}(t_0, x_0, y_0) & \frac{\partial^2 F}{\partial x_2 \partial t}(t_0, x_0, y_0) \end{pmatrix} = \begin{pmatrix} 2(x_0 - \gamma_x(t_0)) & 2(y_0 - \gamma_y(t_0)) \\ -2\gamma'_x(t_0) & -2\gamma'_y(t_0) \end{pmatrix}.$$

This matrix is equivalent to the one for a constant radius function r_c . Therefore, computing the determinant yields the same result:

$$\det(\mathcal{M}_F(t_0, x_0, y_0)) = 4 \cdot \left\langle \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \gamma(t_0), (\gamma'(t_0))^{\perp} \right\rangle.$$

We insert the coordinates $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = P_{\pm}(t_0)$ of the points of regression (t_0, x_0, y_0) and get

$$\begin{aligned} \frac{1}{4} \det(\mathcal{M}_F) &= \left\langle \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \gamma(t_0), (\gamma'(t_0))^{\perp} \right\rangle = \\ &= \left\langle -\frac{r(t_0)r'(t_0)}{\|\gamma'(t_0)\|^2} \gamma'(t_0) \pm \frac{r(t_0)}{\|\gamma'(t_0)\|^2} \sqrt{\|\gamma'(t_0)\|^2 - r'(t_0)^2} (\gamma'(t_0))^{\perp}, (\gamma'(t_0))^{\perp} \right\rangle = \\ &= -r(t_0)r'(t_0) \left\langle \frac{\gamma'(t_0)}{\|\gamma'(t_0)\|}, \frac{\gamma'(t_0)^{\perp}}{\|\gamma'(t_0)\|} \right\rangle \pm r(t_0) \sqrt{\|\gamma'(t_0)\|^2 - r'(t_0)^2} \left\langle \frac{\gamma'(t_0)^{\perp}}{\|\gamma'(t_0)\|}, \frac{\gamma'(t_0)^{\perp}}{\|\gamma'(t_0)\|} \right\rangle \\ &= -r(t_0)r'(t_0) \langle T(t_0), N(t_0) \rangle \pm r(t_0) \sqrt{\|\gamma'(t_0)\|^2 - r'(t_0)^2} \langle N(t_0), N(t_0) \rangle \\ &= r(t_0) \sqrt{\|\gamma'(t_0)\|^2 - r'(t_0)^2}. \end{aligned}$$

This value is non-zero because the radius is always greater than zero and because Equation (2.2) holds. Hence, F is a universal unfolding and the points of regression with A_2 singularity are diffeomorphic to ordinary cusps even in the case of a non-constant radius function r and a non arc-length parameterized curve γ .

Proposition 2.6.8 (A_2 singularities are cusps)

Let \mathcal{F} be the family of curves containing all circles $F(t, x, y) = \|(x, y)^T - \gamma(t)\|^2 - r(t)^2$ for a curve $\gamma: I \rightarrow \mathbb{R}^2$ and a radius function $r: I \rightarrow \mathbb{R}$ defining a stroke. Then the discriminant envelope E_d of \mathcal{F} has an ordinary cusp at an A_2 singularity.

2.6.2 $A_{\geq 3}$ singularities and a connection to catastrophe theory

So far, we have shown that envelopes have cusps at points of regression with an A_2 singularity. Now, we study envelopes with an $A_{\geq 3}$ singularity at a point of regression (t_0, x_0, y_0) . For these points, it holds that in addition to $F = \frac{\partial F}{\partial t} = \frac{\partial^2 F}{\partial t^2} = 0$, the third derivative of F with respect to t is zero, i.e., $\frac{\partial^3 F}{\partial t^3} = 0$.

The main results of this section will be a classification of $A_{\geq 3}$ singularities and a connection of strokes to swallowtail surfaces. The connection of strokes to swallowtail surfaces will be essential for the development of algorithmic ideas on how to conformally map a tiled strip to a stroke with A_2 singularities on the boundary. We will conjecture, however, that $A_{\geq 3}$ singularities are negligible in the sense that our algorithm does not need to treat them differently than regular envelope points (see Section 4.3.2).

As in the previous section, we start with a stroke s defined by a curve γ and a constant radius r_c . Then, the envelope curves of s consist of two parallel curves $\gamma_{\parallel \pm r_c}$ defined in Section 2.3.1 by

$$\gamma_{\parallel \pm r_c}(t) = \gamma(t) \pm \frac{r_c}{\|\gamma'(t)\|} \gamma'(t)^\perp.$$

By Theorem 2.3.3, the parallel envelope has a point of regression (t_0, x_0, y_0) if and only if it has a common point with the evolute ν of γ ,

$$\nu(t) = \gamma(t) + \frac{1}{\kappa(t)\|\gamma'(t)\|} \gamma'(t)^\perp.$$

Therefore, at points of regression, it holds $\pm r_c = \frac{1}{\kappa(t)}$, where κ is the curvature of γ .

For an $A_{\geq 3}$ singularity (t_0, x_0, y_0) of this kind, it holds by Equation (2.26) that the curvature κ of γ has vanishing derivative at t_0 since $\frac{\partial^3 F}{\partial t^3}(t_0, x_0, y_0) = 0$ if and only if $\kappa'(t_0) = 0$. If $\kappa(t_0) \neq 0$, then $\gamma(t_0)$ is an extremum of curvature of γ . If $\kappa'(t_0) = \kappa(t_0) = 0$, then γ is either a straight line, a circle, or has an inflection point at $\gamma(t_0)$. By Section 2.5, strokes with a constant radius of r_c have a boundary with no point of regression when the curvature of γ is zero because the criterion $\pm r_c = \frac{1}{\kappa(t)}$ is never satisfied for finite r_c and $\kappa(t) = 0$. Hence, $A_{\geq 3}$ singularities only occur at points of extreme curvature of γ , provided that the envelope curves are the parallels of γ .

Lemma 2.6.9 For a stroke defined by a curve γ and a constant radius r_c , its envelope curve $\gamma_{\parallel \pm r_c}$ has an $A_{\geq 3}$ singularity at parameter t only if γ has an extremum of curvature there.

In this case, the evolute ν itself has a singularity at (t_0, x_0, y_0) , since the tangent of curve γ changes sign at points of extreme curvature. According to Bruce and Giblin [BG92, p.34], the singular point of the evolute is a cusp. To visualize this, we consider the example of a curve γ that is part of an ellipse, as in Benchmark Example 2.1.9.

Example 2.6.10 Let γ be a part of an ellipse with $a, b > 0$, defined by

$$\gamma(t) = \begin{pmatrix} a \cos(t) \\ b \sin(t) \end{pmatrix} \quad \text{for } t \in \left(\frac{\pi}{2}, \frac{3\pi}{2} \right).$$

For $t \in \left(\frac{\pi}{2}, \frac{3\pi}{2} \right)$, the curvature $\kappa(t)$ of the ellipse has a extremum at $t = \pi$ given by

$$\kappa(t) \Big|_{t=\pi} = \frac{\det(\gamma'(t), \gamma''(t))}{\|\gamma'(t)\|^3} \Big|_{t=\pi} = \frac{ab \sin(t)^2 + ab \cos(t)^2}{\sqrt{(b^2 \cos(t)^2 + a^2 \sin(t)^2)^3}} \Big|_{t=\pi} = \frac{a}{b^2}$$

The derivative of the curvature at $t = \pi$ equals zero:

$$\kappa'(t) \Big|_{t=\pi} = \frac{3ab(-a^2 + b^2) \cos(t) \sin(t)}{\sqrt{(b^2 \cos(t)^2 + a^2 \sin(t)^2)^5}} \Big|_{t=\pi} = 0.$$

Depending on the radius $r_c = \frac{1}{\kappa(t)}$ of the parallel to the curve γ , three distinct cases can be distinguished:

1. For $r_c < \frac{b^2}{a} = \frac{1}{\kappa(\pi)}$, the parallel $\gamma_{\parallel r_c}$ has no points of regression.
2. For $r_c > \frac{b^2}{a} = \frac{1}{\kappa(\pi)}$, $\gamma_{\parallel r_c}$ has two A_2 singularities being cusps (see Corollary 2.6.7).
3. For $r_c = \frac{b^2}{a} = \frac{1}{\kappa(\pi)}$, the point of regression has type $A_{\geq 3}$ since $\kappa'(\pi) = 0$, and the ellipse has an extremum of curvature.

In case 3, the parallel $\gamma_{\parallel r_c}$ for $r_c = \frac{b^2}{a}$ meets the evolute ν of γ precisely at a joint singular point of both curves. All cases are displayed in Figure 2.13 for $a = \cosh(0.3)$ and $b = \sinh(0.3)$.

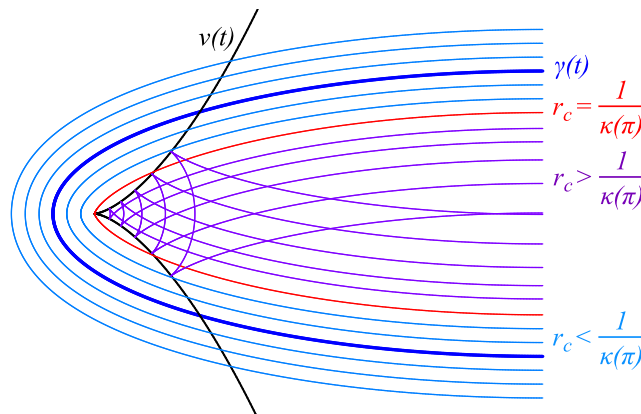


Figure 2.13: A set of parallels $\gamma_{\parallel r_c}$ of an ellipse γ .

To determine the properties of a point of regression with an $A_{\geq 3}$ singularity on a parallel envelope $\gamma_{\parallel r_c}$, we follow Bruce and Giblin [BG92, p.129, iii)] and compare the curves and their parallels to sections with so-called *swallowtail (or dovetail) surfaces*. Figure 2.14 shows a swallowtail surface as defined by Zeeman [Zee77] and Thom [Tho75] from different perspectives. The surfaces themselves give an explanation for their name [Zee77, p.25]: they look like the tails of a swallow (or dove).

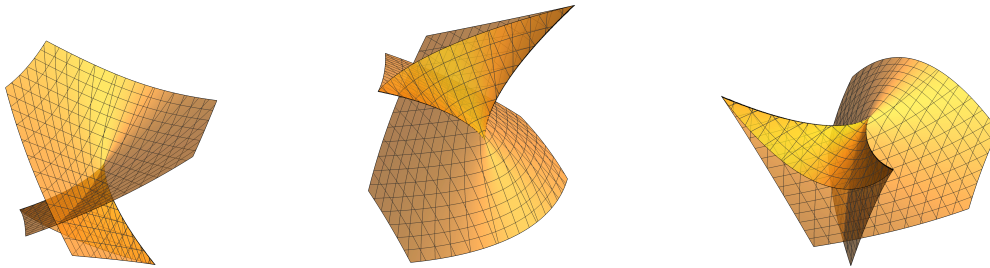


Figure 2.14: A swallowtail surface from different perspectives.

Note 2.6.11 *The notion of a swallowtail comes from catastrophe theory introduced by Thom in the 1960s [Zee79]. Zeeman has further developed catastrophe theory, convinced that it “will prove to be even more far-reaching” [Zee79, p.12]. Thom describes catastrophe theory as the study of discontinuities of a model of a changing system [Tho75, 1.4.A.] A manifold represents the space of observed states, and catastrophes occur when the state changes and the system has a discontinuity. Thom [Tho75] and Zeeman [Zee77] give some applications of catastrophe theory, for example, in the fields of biology, psychology, and physics.*

According to Zeeman [Zee77, p.23 ff.] and Thom [Tho75, p.64f.] a swallowtail surface is the discriminant surface of a smooth 3-dimensional surface M in the 4-dimensional space $C \times X$, where C is a 3-dimensional parameter space, and X is a 1-dimensional state space. The surface M is the set of singular values $\frac{\partial f}{\partial x}(u, v, w, x) = 0$ of a smooth function

$$f(u, v, w, x) = \frac{1}{5}x^5 + ux^2 + vx + w.$$

This surface is the zero set of the 3-dimensional discriminant of $\frac{\partial f}{\partial x}(u, v, w, x) = 0$:

$$0 = 26u^4w - 4u^3v^2 - 128u^2w^2 + 144uv^2w - 27v^4 + 256w^3.$$

In our setting, a swallowtail surface results from the smooth function F defined as above by $F(t, x, y, r) = \|(x, y)^T - \gamma(t)\|^2 - r^2$. The 3-dimensional parameter space consists of points $(x, y, r) \in \mathbb{R}^3$ while the state space consists of parameters $t \in \mathbb{R}$. The swallowtail surface is given by the singular values of $\frac{\partial F}{\partial t}(t, x, y, r)$ for which also $F(t, x, y, r) = 0$. This corresponds to the discriminant envelopes (see Definition 2.2.2) for all $r \in \mathbb{R}$. We call this swallowtail surface \mathcal{S} . It can be deduced from Theorem 2.4.2 and Section 2.3.1 that \mathcal{S} consists of all parallels $\gamma_{\parallel r}(t) = \begin{pmatrix} x_r(t) \\ y_r(t) \end{pmatrix}$ of curve γ for $r \in \mathbb{R}$, lifted to $(x_r(t), y_r(t), r)^T$:

$$\mathcal{S} := \left\{ (x, y, r) \in \mathbb{R}^3 \mid (x, y) = (x_r(t), y_r(t)) \text{ for } \gamma_{\parallel r}(t) = \begin{pmatrix} x_r(t) \\ y_r(t) \end{pmatrix}, t \in I \right\}. \quad (2.30)$$

Figure 2.15 partially shows the surface \mathcal{S} generated by the ellipse $\gamma(t) = \begin{pmatrix} a \cdot \cos(t) \\ b \cdot \sin(t) \end{pmatrix}$ from the previous Example 2.6.10, where $a = \cosh(0.3)$ and $b = \sinh(0.3)$. Curve γ is the intersection of the swallowtail surface \mathcal{S} with the xy -plane at $r_c = 0$ and appears as a dark blue curve in the left picture. Further marked curves are sections of \mathcal{S} corresponding to parallels $\gamma_{\parallel r_c}$ with different kinds of singularities for some $r_c \neq 0$. The color coding in Figure 2.15 corresponds to the color coding in Figure 2.13 for the three types of parallels distinguished in Example 2.6.10.

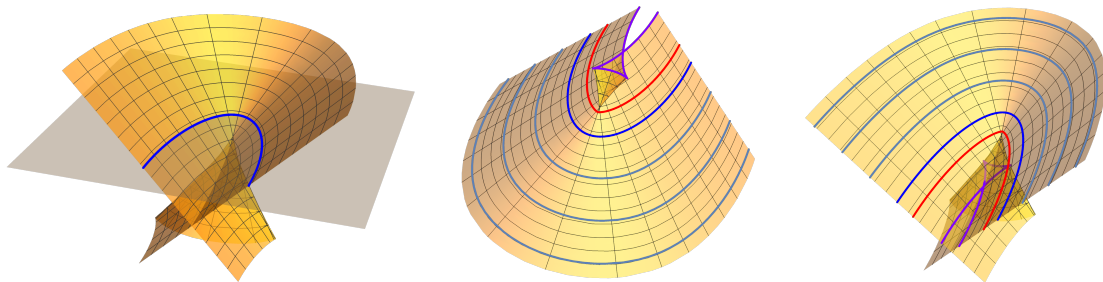


Figure 2.15: Sections of a swallowtail surface with planes for different values of r_c .

By definition of \mathcal{S} in Equation 2.30, the surface consists of all parallels $\gamma_{\parallel r_c}$ of γ lifted to the third dimension by the respective $r_c \in \mathbb{R}$. Hence, if we take the section of \mathcal{S} with the plane at $z = r_c$ parallel to the xy -plane, we obtain the parallel $\gamma_{\parallel r_c}$ as curve of intersection. As a consequence, the sections of \mathcal{S} with two planes at $z = r_c$ and $z = -r_c$ for some $r_c \in \mathbb{R} \setminus \{0\}$ are the lifted envelope curves of γ . If the sections are projected onto the xy -plane, the parallels $\gamma_{\parallel \pm r_c}$ are obtained, which are the envelope curves of the stroke s defined by the curve γ and the constant radius r_c .

There are sections of the surface without singularities corresponding to envelope curves without points of regression for all parameters $t \in I$. This occurs if and only if $\gamma_{\parallel r_c}$ has no common point with the evolute ν of γ , i.e., for all $t \in I$ it holds $r_c \neq \frac{1}{\kappa(t)}$, or equivalently, there is no $t \in I$ such that $\gamma_{\parallel r_c}(t) = \nu(t)$. In Figure 2.15, the light blue sections correspond to the parallels of the ellipse with $r_c < \frac{1}{\kappa(\pi)}$.

Sections of \mathcal{S} and the corresponding parallels $\gamma_{\parallel r_c}$ have cusps if (t_0, x_0, y_0) has an A_2 singularity and the parallel and the evolute of γ meet, i.e., there exists a value $t \in I$ such that $\gamma_{\parallel r_c}(t) = \nu(t)$ (see Theorem 2.3.3). In Figure 2.15, the purple curve represents a section containing two cusps for $r_c > \frac{1}{\kappa(\pi)}$. The cusps are located at the points where the section plane intersects the cuspidal edges of the swallowtail surface. According to Thom [Tho75, p.65], the cuspidal edges of the swallowtail surface are the points where $\frac{\partial f}{\partial x}(a, b, c, x) = 0$ has a triple root in x . In other words, they are the points where $\frac{\partial f}{\partial x} = \frac{\partial^2 f}{\partial x^2} = \frac{\partial^3 f}{\partial x^3} = 0$. This corresponds to the spatial curve $(-3x^4, 8x^3, -6x^2)^T$. The curve is illustrated in green in Figure 2.16a, along with the swallowtail surface. Since all lifted parallels with cusps are sections of a plane with the swallowtail surface \mathcal{S} , where the cusps are points on the cuspidal edges, the projection of the cuspidal edges onto the xy -plane is the locus of the projected cusps of parallels of γ in the plane. By Theorem 2.3.3, the locus of all cusps is the evolute ν of γ . Therefore, the evolute ν is the projection of the cuspidal edges onto the xy -plane. Figure 2.16b displays the surface \mathcal{S} for γ from Example 2.6.10 with its evolute.

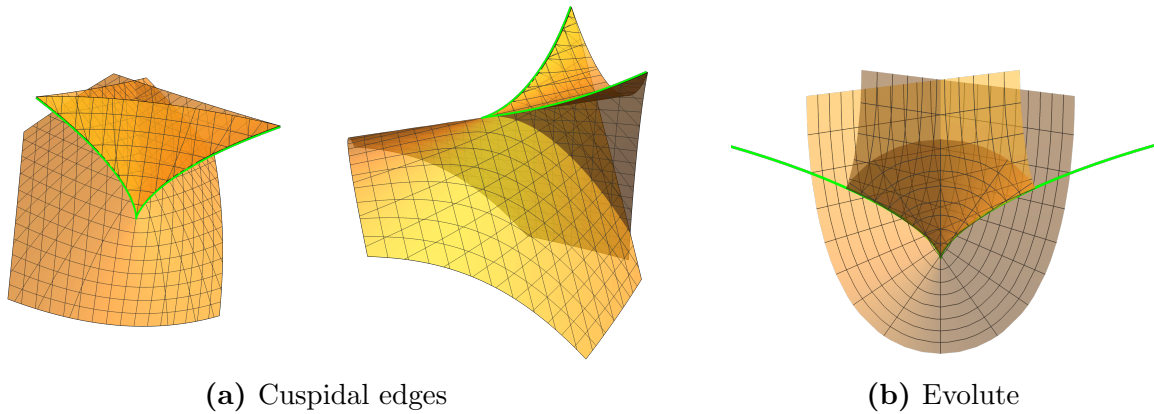


Figure 2.16: Cuspidal edges of the swallowtail surface and the curve’s evolute.

Therefore, the point at which the two cuspidal edges intersect on surface \mathcal{S} coincides with the location of the cusp of the lifted evolute ν on the swallowtail surface. We call this point $z^* \in \mathbb{R}^3$. This point is derived from the singular point (t^*, x^*, y^*) of type $A_{\geq 3}$, where $\nu(t^*)$ has a cusp and $\gamma(t^*)$ has an extremum of curvature $\kappa(t^*)$. At this point z^* , the two singularities on the intersection curve coming from a cuspidal edge collapse to one point. Furthermore, the section of \mathcal{S} with the xy -plane for $r^* = \frac{1}{\kappa(t^*)}$ marks the transition from the section curves with cusps to the regular section curves. In Figure 2.15, the section curve corresponding to $r^* = \frac{1}{\kappa(\pi)}$ is highlighted in red.

According to Thom [Tho75, p.65], z^* on \mathcal{S} has infinite curvature, i.e., the parallel envelope curve $\gamma_{\parallel r^*}(t^*)$ has curvature $\kappa_{r^*}(t^*) = \infty$, where κ_{r^*} is the curvature of $\gamma_{\parallel r^*}$.

This answers the question about the nature of a point on the parallel envelope with an $A_{\geq 3}$ singularity: the parallel has a point of infinite curvature there. From the nature of a swallowtail surface, we deduce that an $A_{\geq 3}$ singularity is locally the only point of infinite curvature of a parallel.

Lemma 2.6.12 (Singular boundary points of strokes with constant radius r_c)

For a stroke that is defined by the curve γ and has a constant radius r_c , the A_2 singularities of the envelope of the stroke are cusps, whereas the $A_{\geq 3}$ singularities are points on the stroke’s envelope that have infinite curvature.

The evolute ν of γ is the locus of all singular points on the envelope and it is the projection of the cuspidal edges of the swallowtail surface \mathcal{S} corresponding to the stroke. The cusp of the evolute ν corresponds to the $A_{\geq 3}$ singularity, which is the projection of the junction of the cuspidal edges on \mathcal{S} .

We will conjecture in Section 4.3.2 that the algorithm that conformally maps a strip to a stroke can neglect $A_{\geq 3}$ singularities. However, the connection between the parallel envelope curves and a swallowtail surface will be very useful if the boundary of a stroke has cusps. It follows from the nature of a swallowtail surface that cusps always arise in pairs, since they originate from a cut through the two cuspidal edges of \mathcal{S} [Tho75, p.65]. Exceptions occur if the stroke begins or ends shortly before or after an extremum of curvature of γ , such that the parallel meets only one of the two cusps. In Sections 4.3.2

and 5.4.2, we will apply this knowledge to develop a scheme for mapping a strip to a stroke when the stroke has cusps on the boundary. The scheme will be based on the projection of the swallowtail surface onto the xy -plane.

So far, we have only analyzed strokes with a constant radius function, resulting in envelope curves parallel to γ . Now we move on to strokes with non-constant radius functions r . We already know that their points of regression of type A_2 are also cusps. However, it is not evident that a swallowtail surface related to γ exists, which would result in sections projected to the xy -plane corresponding to the envelope curves $w_{\pm}(t) = (w_{\pm}^x(t), w_{\pm}^y(t))^T$.

Figure 2.17 indicates that the lifted envelope curves $w_{\pm}^{\ell}(t) = (w_{\pm}^x(t), w_{\pm}^y(t), \pm r(t))^T$ for $t \in I$ typically do not lie on the swallowtail surface \mathcal{S} of γ defined in Equation (2.30) for parallels. The figure displays the swallowtail surface \mathcal{S} from different perspectives, which consists of the blue curve $\gamma(t) = \begin{pmatrix} t-1 \\ 3(t-1)^2 \end{pmatrix}$ for $t \in [0, 2]$ from Benchmark Example 2.1.10 and of its parallels. The red envelope curve $w_{+}^{\ell}(t)$ as defined in Equation (2.13) for $r(t) = 1.4 \cdot \cos(\frac{t}{4} - 1)^2$ is not situated on \mathcal{S} . A computational proof of this is given by showing that there is no real parameter t for which the point $w_{+}^{\ell}(1)$ lies on the section at $r_c = r(1)$ with the surface \mathcal{S} :

$$\nexists t \in \mathbb{R} : w_{+}^{\ell}(1) = \begin{pmatrix} x_{r(1)}(t) \\ y_{r(1)}(t) \\ r(1) \end{pmatrix} \text{ for } \gamma_{\parallel r(1)} = \begin{pmatrix} x_{r(1)}(t) \\ y_{r(1)}(t) \end{pmatrix}.$$

We use Mathematica [WR23] to search for real parameters t where $(w_{+})_x(1) \stackrel{*}{=} x_{r(1)}(t)$ and $(w_{+})_y(1) \stackrel{\circ}{=} y_{r(1)}(t)$. The Mathematica function `Solve` returns $t = 0.950746$ for the equation \star and $t \in \{0.615878, 0.97107, 1.02893, 1.38412\}$ for the equation \circ . Since there is no common root, it follows that the point $w_{+}^{\ell}(1)$ on the lifted envelope does not lie on $\mathcal{S}_{r(1)}$.

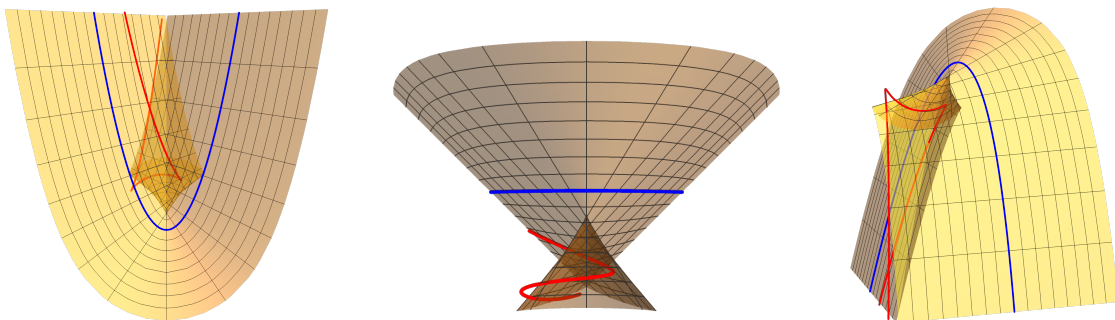


Figure 2.17: The lifted red envelope curve for non-constant radius does not lie on \mathcal{S} .

Thus, we need to define a surface \mathcal{S}_r for a non-constant radius function r such that the envelope curves w_{\pm} are projected sections of \mathcal{S}_r . We define it analogously to the definition of \mathcal{S} . For this, we reformulate the definition of \mathcal{S} in (2.30) as a surface \mathcal{S}_{r_c} between two lifted parallels $\gamma_{\parallel r_c}$ and $\gamma_{\parallel -r_c}$ for a constant radius $r_c \neq 0$:

$$\mathcal{S}_{r_c} := \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \left| \begin{array}{l} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \gamma_{\parallel r_c}(t) + (1-\lambda) \gamma_{\parallel -r_c}(t) \\ z = \lambda r_c + (1-\lambda)(-r_c) \end{array} \right. , \lambda \in [0, 1], t \in I \right\}. \quad (2.31)$$

Replacing $\lambda \in [0, 1]$ with $\lambda \in \mathbb{R}$ restores the definition of \mathcal{S} for an arbitrary $r_c \neq 0$. The surface \mathcal{S}_r between the two envelope curves w_+ and w_- for the curve γ and a non-constant radius function r is defined analogously to \mathcal{S}_{r_c} as

$$\mathcal{S}_r := \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \left| \begin{array}{l} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda w_+(t) + (1 - \lambda)w_-(t) \\ z = \lambda r(t) + (1 - \lambda)(-r(t)) \end{array} \right. , \lambda \in [0, 1], t \in I \right\}. \quad (2.32)$$

Figure 2.18 shows the surface \mathcal{S}_r from different perspectives for $\gamma(t) = (t - 1, 3(t - 1)^2)^T$ and $r(t) = \cos(t/4 - 1)^2$ as in Benchmark Example 2.1.10. The blue line represents γ , while the two red lines correspond to w_{\pm} . The picture in the middle shows that \mathcal{S}_r has no cuspidal edges, although the picture on the left shows that from a perspective parallel to and above the xy -plane, the folds of the surface still appear to be cuspidal edges. We showed in Section 2.6.1 that the singularities of the envelope are indeed cusps. Thus, the two folds can be seen as two independent but merging cusp-catastrophes, which are defined as the zero set of the derivative of $f(a, b, x) = \frac{1}{4}x^4 - ax - \frac{1}{2}bx^2$, i.e., by $\frac{\partial f}{\partial x}(a, b, x) = 0$ [Zee77, p.27, Tab.3], [Zee77, p.6, Fig.4].

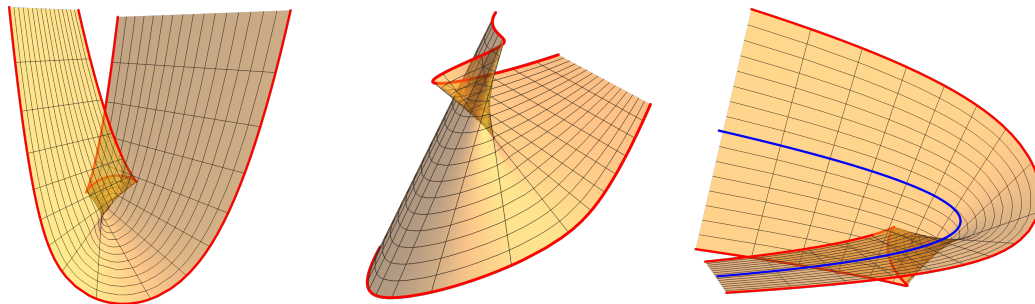


Figure 2.18: Surface \mathcal{S}_r for the stroke from Benchmark Example 2.1.10.

The right picture reveals that the curve γ itself does not lie entirely on the surface \mathcal{S}_r . But this is not problematic, because for our model of strokes only the projections of the surfaces \mathcal{S}_r and \mathcal{S}_{r_c} are important.

The characteristics of the projection of \mathcal{S}_r and \mathcal{S}_{r_c} are related to the location of points of regression on the stroke's envelopes. For strokes with constant radius r_c , cuspidal edges and $A_{\geq 3}$ singularities with infinite curvature only occur around points of extreme curvature of γ , i.e., when $\kappa(t) \neq 0$ and $\kappa'(t) = 0$. For strokes with a non-constant radius function r , singular boundary points are not restricted to occur around curve points of extreme curvature as in Section 2.5. It is also possible that $\kappa = 0$, and thus γ is a straight line, a circle, or has an inflection point.

To date, our research only covers the case of the $A_{\geq 3}$ singularity in the Benchmark Example 2.1.10, which is associated with an extremum of curvature of the curve γ and which never has zero curvature in the considered parameter interval I . The entire range of possibilities in which the envelopes of strokes of non-constant radius may encounter cusps or $A_{\geq 3}$ singularities is left to future research.

The projection of a swallowtail surface around a curve point with extreme curvature maps the cuspidal edges/folds to two- and threefold regions in the stroke. Both projections of Examples 2.6.10 and 2.1.10 are shown in Figure 2.19.

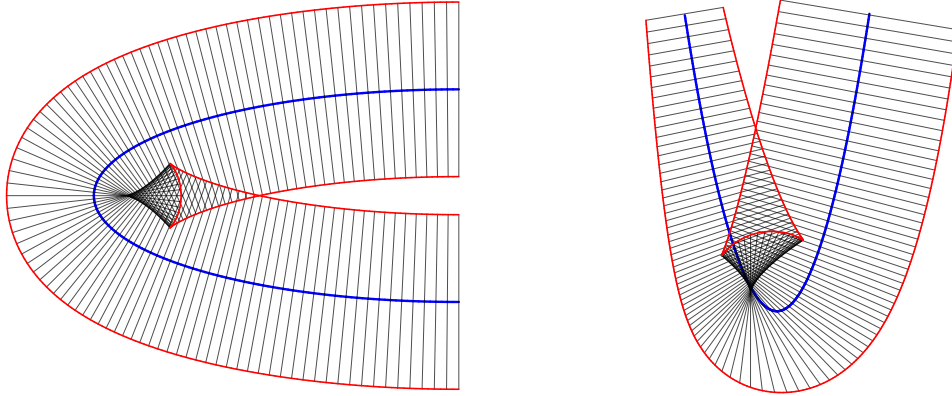


Figure 2.19: Projection of the surfaces \mathcal{S}_{r_c} (Example 2.6.10) and \mathcal{S}_r (Example 2.1.10).

Figure 2.20 shows an example of how these regions arise: a swallowtail surface folds for the first time when the envelope curve meets its first cusp and changes direction. When the curve of the envelope reaches the second cusp, it changes direction so that it corresponds to the direction of the blue curve γ again, causing the surface to fold a second time.

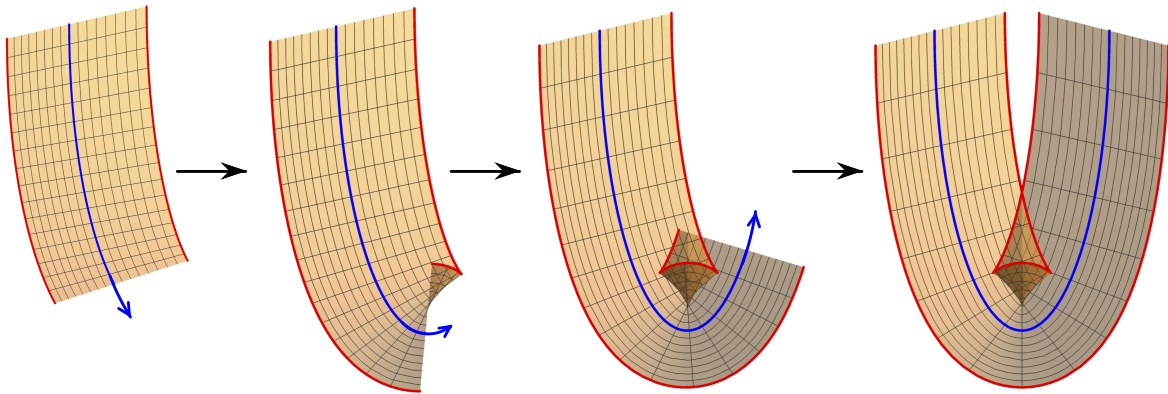


Figure 2.20: Surface \mathcal{S}_{r_c} evolving for γ from Example 2.6.10 and $r_c = \frac{1}{4}$ rotated by $\frac{\pi}{2}$.

This results in three different regions in the projection of the swallowtail surface: a twofold region, a threefold region, and a region without multiple layers. The projections of \mathcal{S}_r and \mathcal{S}_{r_c} in Figure 2.19 both show the envelope curves in red and the curve γ in blue. By definition of \mathcal{S}_r and \mathcal{S}_{r_c} in the respective Equations (2.32) and (2.31), the projected surfaces consist of all points that are a linear interpolation between two respective boundary points $w_+(t)$ and $w_-(t)$ of the same parameter $t \in I$.

In Sections 4.3.2 and 5.4.2, we will revisit the projected swallowtail surfaces near points of extreme curvature on γ . We will discuss how to detect the different regions of one, two, or three layers when drawing a stroke, and how these regions can be used to find a map from the ornamental strip to the stroke in the case of singular boundary points.

3 | Conformality

The aim of our research is to compute a real-time conformal map from a tiled ornamental strip to a stroke drawn by the user. In the previous chapter, we extensively discussed the continuous stroke model, which provided us with a mathematical understanding of the stroke region to which we want to conformally map the tiled strip. We already defined conformal maps in Definition 1.1.2 as maps that preserve oriented angles between curves and infinitesimal shapes, i.e., as local similarity transformations. But there are several equivalent definitions of continuous conformal maps given by Crane [Cra19]: conformal maps are angle preserving, they preserve infinitesimal circles, they are analytic maps, they can be written as the sum of conjugate harmonic functions, and even more. These different definitions of conformality will be important again in Chapter 4.

In this chapter, we will examine the strip and the stroke as simply connected planar domains. We will study properties of existing results on conformal maps between them and their relevance to our task. In Section 3.1, we will see that the Riemann mapping theorem ensures the existence of a conformal map between the strip and the stroke. If three points are specified on the boundary of the strip to map to three points on the stroke's boundary, a conformal map uniquely extends to the boundaries of the domains. In the interior, the map is also unique.

In Section 3.2, we will investigate conformal maps between quadrilaterals with four designated vertices and rectangles, where the vertices are mapped to the corners of the rectangle. These conformal maps exist only if the rectangle has a certain side-length-ratio, known as the conformal modulus. The conformal modulus characterizes conformal equivalence classes of quadrilaterals whose respective conformal image rectangles are all similar. We will examine the properties of the conformal modulus in order to determine the length of the rectangular strip that can be conformally mapped to a stroke. We will see that the conformal modulus must become larger as the stroke gets longer. Moreover, we will investigate the exact conformal map of a half annulus to a rectangle. This map will serve as an accuracy test for our algorithm in Chapter 5.

In Section 3.3, we will study a generalization of the Schwarz reflection principle according to which a conformal map can be extended beyond the boundary of a domain. We will see that, for a boundary consisting of analytic curves, points located infinitesimally close to the boundary outside the domain can be reflected at the local tangent to the curve. The reflected point is located inside the domain, and it is possible to compute its conformal image. Afterwards, the resulting image is reflected again at the boundary of the target region. The resulting point outside the target domain is defined to be the conformal image of the point located outside the preimage domain. In Section 4, we will

show that the envelopes of our strokes consist of analytic curves for at least C^2 -continuous curve γ and radius function r . Hence, this method can be applied for our strokes.

3.1 Conformality of Jordan domains

The stroke and the strip are bounded regions. The strip's boundary is a rectangle consisting of two straight line segments parallel to the x -axis and the two perpendicular line segments parallel to the y -axis that join the upper and lower line segments. This rectangle is the topological image of a circle and, thus, a *Jordan curve* in accordance with the definition by Lehto and Virtanen [LV73, Section 1.3].

The stroke's boundary was discussed in Chapter 2. We saw that the theory about envelopes of a family of circles allows us to characterize the boundary of a stroke in detail. If the stroke is free of self-intersections and the envelopes have no singularities, the stroke is also bounded by plane Jordan curves, namely by the envelope together with the circular arc segments contained in $\mathcal{C}(0)$ and $\mathcal{C}(T)$ at the beginning and end of the stroke, as defined in Definition 2.2.5. For now, we assume that the stroke does not have any self-intersections or singularities. The special cases will be covered in Sections 4.3 and 5.4.

Thus, the stroke and the strip are *Jordan domains*, i.e., they are simply connected planar domains bounded by Jordan curves [LV73, 1.4].

From the Riemann mapping theorem it follows that all Jordan domains can be conformally mapped to each other:

Theorem 3.1.1 (Riemann mapping theorem) [LV73, 2.1]

Every open simply connected domain of the plane that is neither empty nor the entire plane can be conformally mapped onto the open unit disc.

By taking a detour via the unit disk, all Jordan domains are conformally equivalent. However, the very powerful Riemann mapping theorem is not constructive, and we have to investigate further to find out more than existence.

Under the assumption that the preimage and image regions are both simply connected Jordan domains, we can apply a theorem that has various names, for example Osgood-Carathéodory theorem [GS16, Section 2], Carathéodory's theorem [Car98, Section 137] or Theorem on correspondence of boundaries [LV73, Section 2.2].

Theorem 3.1.2 (Carathéodory's theorem)

Every conformal map between simply connected Jordan domains, which are bounded by Jordan curves, can be extended to a homeomorphism of the boundaries, i.e., to a bijective continuous map on the closed domains with continuous inverse.

If three specified points on the preimage and image boundary are mapped to each other, this map is even unique if the triples of points are positively oriented on the Jordan curves. Three points p_1, p_2, p_3 are positively oriented on a Jordan curve if there is a homeomorphism h of the Jordan curve to the unit circle for which the arguments of the image points increase in a range of 2π . This means that for three points p_1, p_2, p_3 on the Jordan curve, the angles $\alpha_i = \arg(h(p_i))$ for $i \in \{1, 2, 3\}$ satisfy $\alpha_1 < \alpha_2 < \alpha_3 < \alpha_1 + 2\pi$

[LV73, Section 1.4]. The extended conformal map uniquely maps the designated oriented points on the preimage Jordan curve to those on the boundary of the image domain.

Proposition 3.1.3 (Uniqueness of conformal extension) [LV73, 2.2]

Let D and D' be two Jordan domains, and let p_1, p_2, p_3 and q_1, q_2, q_3 be positively oriented on the bounding Jordan curves δD and $\delta D'$, respectively. Then there exists a uniquely defined conformal map between the domains, which maps the boundary points p_i to the boundary points q_i for $i = 1, 2, 3$.

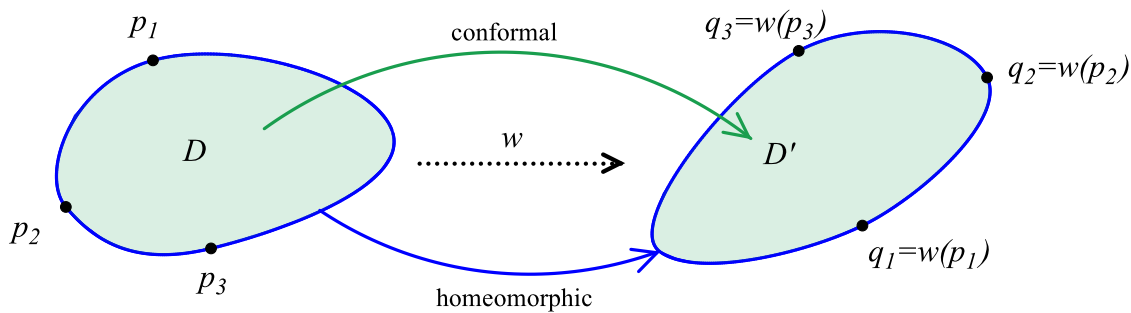


Figure 3.1: Extended conformal map w determined by triples of boundary points.

We call the extended conformal map $w: \overline{D} \rightarrow \overline{D}'$. Figure 3.1 gives a visualization of map w . It conformally maps the open Jordan domain D to D' while, on the boundaries, it is a homeomorphism. It is unique due to two dedicated triples of positively oriented points on the boundaries that are mapped to each other.

This means that the conformal map between the rectangular strip and the stroke can be uniquely extended to the boundaries of three points on the rectangle's and the stroke's boundaries are respectively selected to be mapped to each other. The question is how to choose these points and what effect this choice has on the resulting map.

3.2 Quadrilaterals and conformal modulus

It is intuitive to choose three of the corner points of the rectangular strip as preimage points. On the stroke's boundary, it is reasonable to choose three of the four points where the envelope curves meet the circular arcs of $\mathcal{C}(0)$ and $\mathcal{C}(T)$, i.e., the first and last circles of the stroke. See Figure 2.5 for a reminder.

From Proposition 3.1.3, we know that the mapping is unique as soon as three corners and their respective image points are chosen. Hence, it is not possible to choose where the fourth corner of the rectangle will be mapped to once the three image points are fixed.

So far, we did not specify the length of the tiled strip that is mapped to the stroke. Figure 3.2 shows two versions of Benchmark Example 2.1.7, and it is evident that the length of the strip has a significant impact on the result. The strip mapped to the stroke in the left picture was half as long as the preimage strip in the right picture.

In conformal mapping theory, it is well known that every Jordan domain with four specified vertices belongs to a specific conformal equivalence class. The length-ratio of

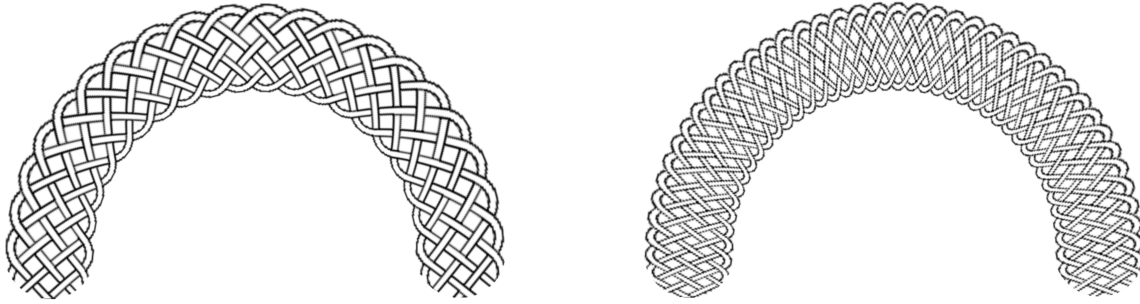


Figure 3.2: Two ornamental strips with different lengths mapped to the same stroke.

the rectangle that the domain can be conformally mapped to in such a way that domain vertices map to rectangle vertices defines this equivalence class. To examine the applicability of this theory to our strokes and rectangular strips, we introduce quadrilaterals and conformal moduli.

Definition 3.2.1 (Quadrilaterals) [LV73, 2.3]

Let W be a Jordan domain bounded by a Jordan curve ∂W , and let z_1, z_2, z_3, z_4 be four positively-oriented points on ∂W . Then the point set $Q(z_1, z_2, z_3, z_4)$, consisting of all points within the closed Jordan domain \overline{W} , is called a quadrilateral. The points z_1, z_2, z_3, z_4 are called vertices. They divide the quadrilateral's boundary ∂W into four Jordan arcs which are called the sides of $Q(z_1, z_2, z_3, z_4)$. To distinguish pairs of "opposite" sides, the arcs connecting z_1 with z_2 and z_3 with z_4 are called a -sides, and the arcs connecting z_2 with z_3 and z_4 with z_1 are called b -sides.

As we already know, it is not possible to map any two quadrilaterals conformally to each other since the image of the fourth vertex is already determined by the first three image-preimage pairs. For this reason, the quadrilaterals are divided into conformal equivalence classes [LV73, 2.3].

Definition 3.2.2 (Conformal equivalence class)

A conformal equivalence class contains all quadrilaterals $Q(z_1, z_2, z_3, z_4)$ that can be conformally mapped to each other.

To put a finger on which quadrilaterals are in one equivalence class, we investigate the conformal map m of a quadrilateral $Q(z_1, z_2, z_3, z_4)$ to a rectangle which maps the vertices of Q to the corners of the rectangle. We follow some intermediate steps [LV73, 2.4], [Con19]. Figure 3.3 presents a visual overview of the entire map m .

First, the quadrilateral $Q(z_1, z_2, z_3, z_4)$ is mapped onto the unit disk D . The Riemann mapping theorem 3.1.1 states that there is a map $F: Q(z_1, z_2, z_3, z_4) \rightarrow D$ that conformally maps Q to D . By Proposition 3.1.3, this map F is even unique if we specify that

$$F(z_2) = -i, \quad F(z_3) = i, \quad F(z_4) = -1.$$

Then, z_1 is mapped to some point w on the unit circle between -1 and $-i$ because the conformal mapping preserves the orientation of vertices on the boundaries.

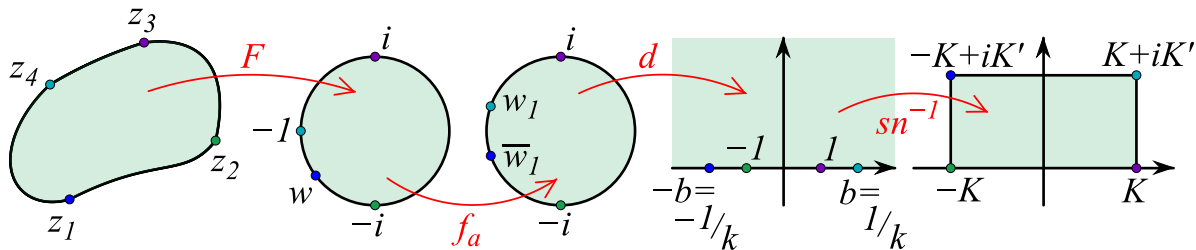


Figure 3.3: Unique conformal map m of a quadrilateral to a rectangle.

Second, we define an automorphism of the disk with fixed points i and $-i$ that maps -1 and w to complex conjugate points w_1 and $\overline{w_1}$, respectively. This is achieved by the Möbius transformation

$$f_a(z) = \frac{z - ia}{1 + ia z}$$

where $a \in (-1, 0)$. The transformation f_a maps -1 to a point between -1 and i on the unit circle ∂D . There exists a unique value of a for which $f_a(-1) = w_1$ and $f_a(w) = \overline{w_1}$, since Möbius transformations leave cross-ratios of four points on a circle invariant [RG11, p.316]. This value of $a \in (-1, 0)$ depends on the location of $F(z_1) = w$ on the boundary of D and, consequently, on the position of z_1 in Q . Once w is known, it holds

$$(i, -1; w, -i) \stackrel{!}{=} (i, w_1; \overline{w_1}, -i)$$

where the left value is a known constant and the right cross-ratio has one unknown variable $\varphi \in (\pi/2, \pi)$, given $w_1 = e^{i\varphi}$ and $\overline{w_1} = e^{i(2\pi-\varphi)}$. This equation can be uniquely solved for φ .

Next, we use a fractional linear map d to map the unit disk D to the upper half plane \mathbb{H}_+ :

$$d(z) = i \cdot \frac{1 - z}{1 + z}.$$

This map d maps 1 to 0 and $\pm i$ to ± 1 . Hence, the right part of the unit circle between $-i$ and i is mapped to the real line between -1 and 1 , while the left part is symmetrically mapped to the real intervals $(-\infty, -1)$ and $(1, \infty)$, respectively. The point -1 is mapped to infinity. Hence, there is a real constant $b > 1$ for which

$$d(\overline{w_1}) = -b, \quad d(-i) = -1, \quad d(i) = 1, \quad d(w_1) = b.$$

As a last step, we replace b with $\frac{1}{k}$ for consistency with the literature. To map the upper half plane to a rectangle, we use elliptic functions. According to [LV73, 2.4] and [Neh75, p.280 ff.], the appropriate conformal map is given by the inverse of the *Jacobi elliptic function* $\operatorname{sn}(z, k)$ also written as $\operatorname{sn}_k(z)$:

$$\operatorname{sn}_k^{-1}(z) = \operatorname{sn}^{-1}(z, k) := \int_0^z \frac{ds}{\sqrt{(1-s^2)(1-k^2s^2)}}.$$

It conformally maps the upper half plane to a rectangle with corners $-K$, K , $K + iK'$ and $-K + iK'$ where

$$K = \int_0^1 \frac{ds}{\sqrt{(1-s^2)(1-k^2s^2)}}, \quad K' = \int_0^1 \frac{ds}{\sqrt{(1-s^2)(1-(1-k^2)s^2)}}. \quad (3.1)$$

Thereby, it holds

$$\begin{aligned} \operatorname{sn}^{-1}(-1/k, k) &= -K + iK', & \operatorname{sn}^{-1}(1/k, k) &= K + iK', \\ \operatorname{sn}^{-1}(-1, k) &= -K, & \operatorname{sn}^{-1}(1, k) &= K. \end{aligned}$$

For further details on the Jacobi elliptic sine function sn and a specific conformal mapping from an ellipse to an infinite strip, see Appendix A.

Combining the last four steps yields a conformal map from $Q(z_1, z_2, z_3, z_4)$ to the rectangle R , whose corners are $-K + iK', -K, K$ and $K + iK'$. As visualized in Figure 3.3, this conformal map is given by

$$m = \operatorname{sn}_k^{-1} \circ d \circ f_a \circ F: Q(z_1, z_2, z_3, z_4) \rightarrow R \quad (3.2)$$

with

$$m(z_1) = -K + iK', \quad m(z_2) = -K, \quad m(z_3) = K, \quad m(z_4) = K + iK'. \quad (3.3)$$

We can deduce that every equivalence class contains a rectangle [LV73, 2.4], since for every quadrilateral there is a unique conformal map of the above form mapping it to a rectangle. If two quadrilaterals Q_1 and Q_2 are mapped to the same rectangle R by the two conformal maps m_1 and m_2 , respectively, then they belong to the same equivalence class since $m_2^{-1} \circ m_1$ conformally maps Q_1 to Q_2 and $m_1^{-1} \circ m_2$ does the same vice versa. Additionally, if a conformal map exists between two rectangles R_1 and R_2 , all quadrilaterals that can be conformally mapped to one of them are in the same class.

According to [LV73, 2.4] and [RG11, p.225], similarity transformations preserve angles and are therefore conformal maps. Consequently, similar rectangles are in the same equivalence class. Moreover, no other conformal maps exist between rectangles, except for similarity transformations. To prove this, we adopt the argument from [Dap18].

Let rectangles R_1 and R_2 have side lengths a_1, b_1 and a_2, b_2 , respectively. Assume that R_1 and R_2 are not similar, so $a_1/b_1 \neq a_2/b_2$. Further assume that there is a conformal map $f: R_1 \rightarrow R_2$. We can then apply the Schwarz reflection principle, also known as the symmetry principle, inversion principle, or reflection principle by Riemann and Schwarz.

Theorem 3.2.3 (Schwarz reflection principle) [Neh75, p.184]

Let D_z and D_w be two domains whose boundaries contain circular arcs or line segments α_z and α_w , respectively. If $f: D_z \rightarrow D_w$ is conformal and maps α_z to α_w , then f can be continued along α_z and α_w as follows. Let the domain D_z^ emerge from D_z by (circle) inversion with respect to the (circular arc or) line segment α_z . If $z \in D_z$ and $z^* \in D_z^*$ are inverse points with respect to the (circle) inversion across α_z , then the image point $w^* := f(z^*) \in D_w^*$ is defined to be the inverse of $w = f(z) \in D_w$ with respect to the corresponding (circle) inversion across α_w and $f: D_z \cup D_z^* \rightarrow D_w \cup D_w^*$ is conformal.*

We use the Schwarz reflection principle to mirror the two rectangles R_1 and R_2 along corresponding sides. This results in a conformal map for larger rectangles, and we may continue this process until the entire plane is covered by reflections of the rectangles R_1 and R_2 , respectively. Hence, we have a conformal automorphism of the entire plane

which is a Möbius transformation [Bea85]. Möbius transformations preserve cross-ratios [Car98, Section 10], and therefore, the cross-ratios of the four corners of the rectangles R_1 and R_2 must be the same. Without loss of generality, assume that R_1 has vertices $0, a_1, a_1 + ib_1, ib_1$ and R_2 has vertices $0, a_2, a_2 + ib_2, ib_2$, listed counterclockwise. If the cross-ratios are equal, then it holds that

$$\begin{aligned} (0, a_1; a_1 + ib_1, ib_1) &= (0, a_2; a_2 + ib_2, ib_2) \\ \frac{-(a_1 + ib_1)(a_1 - ib_1)}{(-ib_1)^2} &= \frac{-(a_2 + ib_2)(a_2 - ib_2)}{(-ib_2)^2} \\ \left(\frac{a_1}{b_1}\right)^2 + 1 &= \left(\frac{a_2}{b_2}\right)^2 + 1 \end{aligned}$$

which is not possible since $a_1/b_1 \neq a_2/b_2$. Therefore, it follows that there are no conformal mappings between rectangles that are not similar. Consequently, only similar rectangles belong to the same equivalence class.

Proposition 3.2.4 *A conformal equivalence class of quadrilaterals $D(z_1, z_2, z_3, z_4)$ contains only those quadrilaterals that can be conformally mapped to similar rectangles. In other words, two quadrilaterals D_1 and D_2 are in a conformal equivalence class if and only if the corresponding conformal map m in Equation (3.2) for D_1 and D_2 maps them to similar rectangles R_1 and R_2 , which are also part of the conformal equivalence class.*

Therefore, all rectangles in a one conformal equivalence class have the same ratio of side lengths [LV73, 2.4]. Hence, the ratio is uniquely determined for each conformal equivalence class and is an important invariant called the *conformal modulus*.

Definition 3.2.5 (Conformal modulus) [LV73, 2.4]

The ratio of the side lengths a/b of rectangles in a conformal equivalence class is called conformal modulus. It is denoted by $M(Q)$, where Q is a quadrilateral in the conformal equivalence class.

It follows that two quadrilaterals can be conformally mapped to each other if and only if they have the same modulus. This provides us with a criterion for our map from a tiled strip to a stroke. The ratio of the side lengths of the strip must be equal to the conformal modulus of the stroke if we want the corners of the rectangular strip to be mapped onto the four points in the stroke boundary where the envelope curves w_{\pm} meet the circles $\mathcal{C}(0)$ and $\mathcal{C}(T)$. Therefore, in order to determine the appropriate length of the strip, it would be necessary to know the modulus of the stroke.

Note 3.2.6 *Sometimes in the literature, the conformal modulus is defined as the ratio b/a [PS10, Chapter 2.1], [Hen93, Chapter 16.11]. This depends on the conformal mapping used to map the quadrilateral to the upper half plane and further the half plane to the rectangle. Nevertheless, it remains true that the same quadrilaterals are in the same equivalence class, as long as one definition is used consistently.*

To avoid confusion about which are the a - and b -sides of our quadrilateral stroke and the ornamental strip, respectively, we fix the convention that the lower left corner of the



Figure 3.4: Vertices of a rectangular ornamental strip.

strip is z_1 , the lower right corner is z_2 , the upper right corner is z_3 and the upper left corner z_4 (see Figure 3.4). Hence, the conformal modulus of the ornamental rectangular strip is given by $M(s) = \frac{|z_1 - z_2|}{|z_4 - z_1|}$, i.e., the width of the strip divided by its height.

While the conformal modulus of a rectangle is calculated by a simple length-ratio, is not easy to derive the conformal modulus of a general quadrilateral. The conformal modulus of a quadrilateral can be determined uniquely by applying the conformal map m from Equation (3.2), which connects the quadrilateral to the corresponding rectangle in the same conformal equivalence class. However, finding and calculating the conformal map m can be challenging. Therefore, we investigate how to compute the modulus for a given quadrilateral without finding the corresponding rectangle via m .

3.2.1 Properties and computation of the conformal modulus

The conformal modulus of quadrilaterals has several interesting properties, which give insight into the connection between the rectangular strip and a stroke.

First, it is important in which order the vertices z_1, z_2, z_3 and z_4 of quadrilateral $Q(z_i, z_j, z_k, z_\ell)$ with $i, j, k, \ell \in \{1, 2, 3, 4\}$ are considered. According to Definition 3.2.1, the arcs $z_i z_j$ and $z_k z_\ell$ are called a -sides and the arcs $z_j z_k$ and $z_\ell z_i$ are called b -sides for $Q(z_i, z_j, z_k, z_\ell)$. This results in a conformal modulus of $M(Q) = \frac{a}{b}$, where a and b are the side lengths of the rectangle R to which $Q(z_i, z_j, z_k, z_\ell)$ is conformally mapped so that the a - and b -sides are mapped to the sides of the rectangle of length a and b , respectively (see Figure 3.5).

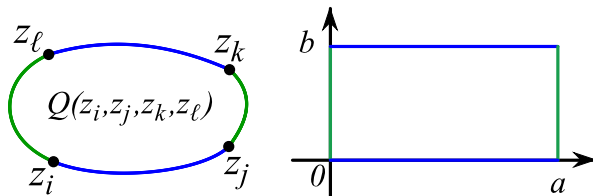


Figure 3.5: The a - and b -sides of Q conformally mapped to R of side lengths a and b .

Hence, the quadrilaterals $Q_1 = Q(z_1, z_2, z_3, z_4)$ and $Q_3 = Q(z_3, z_4, z_1, z_2)$ have the same modulus $M = M(Q_1) = M(Q_3)$, whereas the quadrilaterals $Q_2 = Q(z_2, z_3, z_4, z_1)$ and $Q_4 = Q(z_4, z_1, z_2, z_3)$ have interchanged a - and b -sides with respect to Q_1 and Q_3 . According to Papamichael et al. [PS10, Definition 2.1.4], we call the quadrilaterals Q_2

and Q_4 reciprocal quadrilaterals of Q_1 and Q_3 . Their conformal moduli are the reciprocal of M , namely $M' = \frac{1}{M} = M(Q_2) = M(Q_4)$ [PS10, Theorem 2.1.2].

Additional interesting properties of the conformal modulus of quadrilaterals are based on comparisons of moduli. For instance, consider a quadrilateral $Q(z_1, z_2, z_3, z_4)$ with an additional vertex \tilde{z}_1 on the Jordan curve between z_4 and z_1 (see Figure 3.6 on the left). Then it holds that the modulus $M(Q)$ of Q is smaller than the modulus $M(\tilde{Q})$ of the quadrilateral $\tilde{Q}(\tilde{z}_1, z_2, z_3, z_4)$ containing \tilde{z}_1 as vertex instead of z_1 [PS10, Thm 2.3.4]. This also applies to a vertex \tilde{z}_3 located on side z_2z_3 .

For our purpose, this means that the closer z_1 and z_4 or z_3 and z_2 are, the larger the conformal modulus of the quadrilateral. In other words, the smaller the b -sides are compared to the a -sides of the quadrilateral, the larger the modulus. A third way to state this is that the modulus increases as the stroke gets longer, if the vertices on the boundary of the stroke are chosen to be the four intersections between $\mathcal{C}(0)$, $\mathcal{C}(T)$, and the envelope curves.

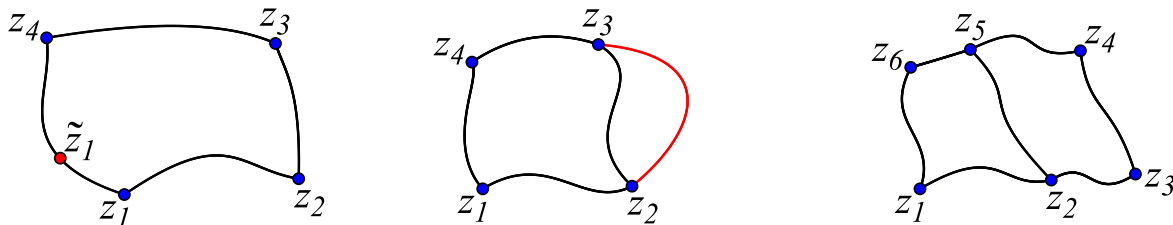


Figure 3.6: Modifications on quadrilaterals and their vertices.

Furthermore, we examine a quadrilateral $Q(z_1, z_2, z_3, z_4)$ with modulus $M(Q)$ and a second quadrilateral $\hat{Q}(z_1, z_2, z_3, z_4)$ that has the same vertices but such that $Q \subset \hat{Q}$. This occurs, for example, when the side z_2z_3 is replaced by another Jordan curve outside Q , as in Figure 3.6 in the middle. Then the modulus $M(\hat{Q})$ is greater than the original modulus $M(Q)$ [Hen93, Thm 16.11i], [PS10, Thm 2.3.5]:

$$M(\hat{Q}) > M(Q).$$

In our case, this means that if the stroke is drawn and the vertices are fixed, and then the stroke is extended by the drawing user, the new stroke will have a larger conformal modulus.

Another property of the conformal modulus is the result of its additivity. When looking at a Jordan region with six distinguished vertices z_1, \dots, z_6 as shown in Figure 3.6 on the right, we can define three quadrilaterals $Q(z_1, z_3, z_4, z_6)$, $Q_1(z_1, z_2, z_5, z_6)$, and $Q_2(z_2, z_3, z_4, z_5)$. According to Papamichael et al. [PS10, Thm 2.3.6] or Henrici [Hen93, Thm 16.11j], it holds that the sum of the moduli $M(Q_1)$ and $M(Q_2)$ is less than or equal to the modulus of the entire quadrilateral $M(Q)$:

$$M(Q_1) + M(Q_2) \leq M(Q).$$

So when the stroke increases, it is not generally possible to add the moduli of the individual parts, as this would not necessarily result in the correct modulus of the entire stroke.

Overall, the modulus increases for a longer stroke. Therefore, when the stroke grows, a longer rectangular strip must be used as preimage for the map.

Knowing some of the properties of the conformal modulus, we investigate how to concretely calculate the modulus of a quadrilateral.

For this, we recall the conformal map m from Equation (3.2). This map is decomposed into a conformal map F mapping $Q(z_1, z_2, z_3, z_4)$ onto the unit disk, a Möbius transformation f_a rearranging the image points of the vertices z_i on the disk, a linear fractional map d mapping the unit disk onto the upper half plane, and finally the inverse of the Jacobi elliptic function $\text{sn}(z, k)$ mapping the upper half plane onto a rectangle R . The vertices of the quadrilateral are mapped by m as given in Equation (3.3):

$$m(z_1) = -K + iK', \quad m(z_2) = -K, \quad m(z_3) = +K, \quad m(z_4) = K + iK'.$$

Hence, the modulus is calculated in terms of the integral values K and K' by

$$M(Q) = \frac{K'}{2K} \stackrel{(3.1)}{=} \frac{\int_0^1 \frac{ds}{\sqrt{(1-s^2)(1-k^2s^2)}}}{\int_0^1 \frac{ds}{\sqrt{(1-s^2)(1-(1-k^2)s^2)}}$$

which is also stated by Papamichael et al. [PS10, Equation (2.4.9)].

The substeps of mapping the quadrilateral Q to the rectangle R by m lead to a conformal image of the quadrilateral in the unit disk. To compute the modulus, it is sufficient to know the images of the vertices of Q being mapped on the unit circle by F and the cross-ratio of the image points $F(z_i)$ for all $i \in \{1, 2, 3, 4\}$.

In our case, $F(z_4) = -1$, $F(z_1) = w = \exp(i\varphi)$ for $\varphi \in (-\pi, -\frac{\pi}{2})$, $F(z_2) = -i$, and $F(z_3) = i$. Hence, their cross-ratio c is given by

$$c = (F(z_4), F(z_1); F(z_2), F(z_3)) = (-1, e^{i\varphi}; -i, i) = -\frac{\cos(\varphi)}{\sin(\varphi) + 1} \in (1, \infty).$$

The modulus of long thin quadrilaterals Q with large a -sides and small b -sides can be approximated by the cross-ratio c by the formula [PS10, Remark 2.5.3]

$$M(Q) \approx -\frac{1}{\pi} \ln \left(\frac{c-1}{16} \right). \quad (3.4)$$

If $c < 17$, then the modulus is in the interval $[0, \infty)$, but if $c > 17$, the modulus is negative. To approximate the modulus, it is necessary to calculate the cross-ratio c in every individual case. In our specific case of mapping a rectangular strip to a stroke, we have to work with an elongated region where z_4z_1 and z_2z_3 are much shorter than z_1z_2 and z_3z_4 . Thus, the image of z_1 under F , i.e., $F(z_1) = w = \exp(i\varphi)$ as above, will be closer to $-1 = F(z_4)$ than to $-i = F(z_2)$. The cross-ratio is thus close to $1 = (-1, e^{-i\pi}; -i, i)$, which is less than 17, so the modulus is positive and quite large since

$$M(Q) \stackrel{(3.4)}{\approx} -\frac{\ln((c-1)/16)}{\pi} \xrightarrow{c \rightarrow 1} -\frac{-\infty}{\pi} = \infty.$$

However, computing F as a conformal map from the quadrilateral to the unit disk is a challenging task. In general, only numerical approximations and similar methods are available to compute this map. We reviewed several approaches to numerical approximation of conformal maps in Section 1.3. However, none of the existing techniques meet our requirement to deal with constantly changing domains, i.e., the user-drawn strokes.

The following approach, which considers conformal maps using conformal moduli, is stated by Palka [Pal75]. Two quadrilaterals can be conformally mapped to each other if and only if they have the same conformal modulus. This means that a map f between two domains D and D' is conformal if and only if the modulus $M(Q)$ of any quadrilateral Q in D is equal to the modulus of its image $f(Q)$ in D' under f : $M(Q) = M(f(Q))$. Therefore, it is necessary to verify for all quadrilaterals within a domain whether the modulus is preserved under f to get a positive answer to the question whether f is conformal. Since the number of verifications is very large, the goal of Palka is to reduce the number of quadrilaterals that need to be considered in this verification.

Palka states [Pal75, Theorem 1] that a function f is conformal if the modulus $M(f(Q))$ is less than or equal to the modulus $M(Q)$ of its preimage for each square Q in D with $M(Q) = 1$ and for every rectangle Q with $M(Q) > 1$, having sides parallel to both the x - and y -axis. Thus, if we could check the preservation of the moduli of all squares and rectangles parallel to the coordinate axis, we could prove that the map at hand is conformal. We will revisit this argument in Section 5.3.1 to demonstrate that our approach results in a nearly conformal mapping from a rectangular strip to a stroke.

3.2.2 Conformal map of an upper half annulus to a rectangle

Up until now, we have only considered general results on the properties of rectangles, quadrilaterals, conformal moduli, and conformal maps. Moving forward, our focus is on examining the explicit conformal map between the upper half annulus A and a rectangle R , as shown in Figure 3.7. The half annulus is very similar to the stroke in Benchmark Example 2.1.7. Hence, we will compare the moduli of the explicitly mapped half annulus A with the length-ratio of the ornamental strip that is mapped to Benchmark Example 2.1.7 by our algorithm later in Section 5.3.3. This method of testing the accuracy of a conformal approximation is also used in [NRR⁺22].

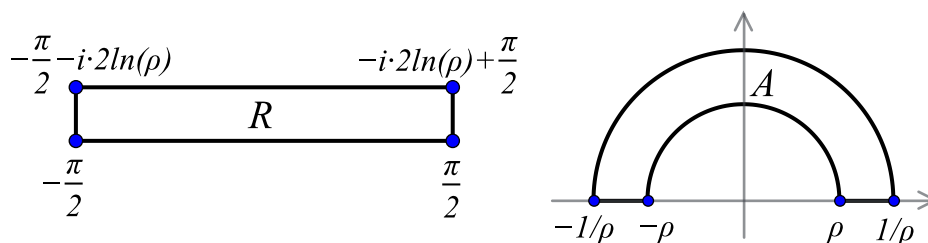


Figure 3.7: A rectangle and its conformal image under $f(w) = i\rho e^{-iw}$: a half annulus.

According to Eskandari [Esk22, p.4], the function

$$f(w) = i\rho e^{-iw} = z = x + iy$$

maps the rectangle R with vertices

$$w_1 = -\frac{\pi}{2}, \quad w_2 = \frac{\pi}{2}, \quad w_3 = \frac{\pi}{2} - i \cdot 2 \ln(\rho) \quad \text{and} \quad w_4 = -\frac{\pi}{2} - i \cdot 2 \ln(\rho)$$

to the open half annulus A with $y > 0$ and $\rho < |z| < \frac{1}{\rho}$ for $\rho \in (0, 1)$. The vertices w_i of R are mapped to the vertices z_i of A as follows:

$$f(w_1) = z_1 = -\rho, \quad f(w_2) = z_2 = \rho, \quad f(w_3) = z_3 = \frac{1}{\rho}, \quad f(w_4) = z_4 = -\frac{1}{\rho}.$$

Since this conformal map f explicitly states that the half annulus A and the rectangle R are conformally equivalent, they have the same conformal modulus. This modulus is defined as the ratio of the side lengths of the rectangle R , namely as

$$M(A) := M(R) = \frac{|w_1 - w_2|}{|w_2 - w_3|} = -\frac{\pi}{2 \ln(\rho)}. \quad (3.5)$$

Since the parameter ρ lies in the interval $(0, 1)$, the logarithm gives a negative value and the modulus is positive.

The half annulus A is bounded by the two half circles $H_k = \{r_k \cdot e^{i\varphi}, \varphi \in (0, \pi)\}$ for $k \in \{1, 2\}$ on the upper half plane around the origin with radii $r_1 = \rho$ and $r_2 = \frac{1}{\rho}$, respectively. Hence, two points $z_k = r_k \cdot e^{i\varphi}$ on the two circles maintain constant distance of $r_2 - r_1 = \frac{1}{\rho} - \rho$ for every $\varphi \in (0, \pi)$. Thus, the stroke s_A that best approximates the half annulus A is a stroke with a constant radius function $r_c = \frac{1}{2} \left(\frac{1}{\rho} - \rho \right)$ and curve $\gamma(t) = \frac{1}{2} \left(\frac{1}{\rho} + \rho \right) \cdot \begin{pmatrix} \cos(\pi - t) \\ \sin(\pi - t) \end{pmatrix}$. This stroke matches the stroke of Benchmark Example 2.1.7 with $t \in I = (0, \pi)$. The half annulus and the stroke are shown in Figure 3.8.

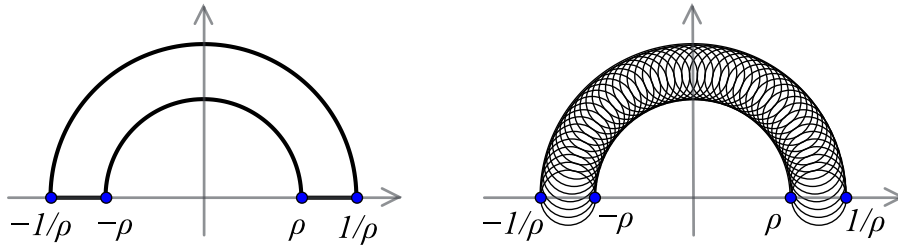


Figure 3.8: A stroke approximating a half annulus.

The stroke s_A and the half annulus A are not exactly the same: the annulus closes the boundary between the two half circles H_1 and H_2 with two line segments from $-\frac{1}{\rho}$ to $-\rho$ as well as from ρ to $\frac{1}{\rho}$. The stroke, however, connects the envelope curves H_1 and H_2 with two lower half circles around $\gamma(0) = \rho + r_c$ and $\gamma(\pi) = -\rho - r_c$, respectively, both of radius r_c and with argument in $(\pi, 2\pi)$. Nevertheless, we can compare the conformal modulus of the half annulus A and of the rectangular strip being the preimage of the stroke s_A in our algorithm. If the moduli are approximately equivalent, we have an indicator that the map is not too far away from a conformal map. In Section 5.3.3 we apply our algorithm to calculate the conformal map for Benchmark Example 2.1.7 and test how well conformality is approximated.

3.3 Extension of conformal maps to the boundary and beyond

In Chapter 2, we have extensively analyzed the boundary of a stroke. Now, we will consider how to extend the conformal map from the rectangular strip to the stroke to the boundaries. For this, it is important to understand how a conformal map extends beyond the boundaries of its source and target domains.

We already introduced the theorem of Carathéodory in Theorem 3.1.2. This theorem states that the conformal map between two Jordan domains can be extended to a homeomorphism of the boundaries. Furthermore, Proposition 3.1.3 states that for three specified pairs of preimage and image points that are mapped to each other, this extension is unique. To extend the map beyond the boundaries, the Schwarz reflection principle 3.2.3 gives a procedure how to define the map outside the Jordan domains bounded by line segments and/or circular arcs. To obtain the image point w of a point z located outside the preimage domain, the point z is inverted with respect to the line segment or circular arc of the boundary. The inverted point z' lies inside the preimage domain and is mapped by f to $w' = f(z')$. Finally, w is the reflection of w' at the boundary of the target domain. A visualization of this can be seen in Figure 3.9.

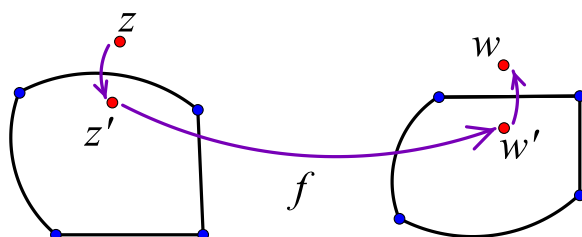


Figure 3.9: Conformal image of point z outside the domain by Schwarz reflection.

However, our stroke domain is not only bounded by line segments and circular arcs. Thus, it is necessary to explore a generalization of the Schwarz reflection principle to a domain bounded by general envelope curves and two circular arcs, as it is the case for our strokes (see Definition 2.2.5).

Needham describes how to imitate reflection on any analytic curve [Nee23, Chapter 5.XI.5, p.252 ff.]. An analytic curve $\delta: \mathbb{R} \rightarrow \mathbb{R}^n$, $\delta(t) = (x_1(t), \dots, x_n(t))$ is a function where each coordinate $x_i(t), i \in \{1, \dots, n\}$ can locally be written as a convergent power series $x_i(t) = \sum_{k=0}^{\infty} a_k(t - c)^k$ for a real constant c and coefficients $a_k \in \mathbb{R}$ [Lan13, p.68], [Nee23, p.228]. We will show in Section 4.1.2 that the envelope curves of the strokes in our algorithm are analytic and that the result for a generalization of the Schwarz reflection principle for analytic curves can be applied.

For an analytic curve δ in the complex plain \mathbb{C} , we call the reflection on this curve $R_\delta: \mathbb{C} \rightarrow \mathbb{C}$. To ensure that the reflection R_δ is a valid generalization of a reflection on a line or circle, we require that $R_\delta(a) = a$ for all points a on the curve δ . For all other

3.3. Extension of conformal maps to the boundary and beyond

points $z \neq \delta(t)$ for some t , the reflection R_δ is a composition of some analytic function S_δ and complex conjugation:

$$R_\delta(z) = \overline{S_\delta(z)}.$$

The analytic function S_δ is called *Schwarz function*. For points a on δ it holds that $R_\delta(a) = a = \overline{\overline{a}} = \overline{S_\delta(a)}$, which implies that the Schwarz function maps these points to their complex conjugates. However, this is not the case for points z not on δ .

We assume that z lies on an infinitesimal circle around $a = \delta(t)$ for some t . The curve δ is conjugated by S_δ as a whole. According to Needham [Nee23, p.255], the image of the infinitesimal circle under S_δ is a composition of scaling (also called amplification) and rotation (also called twist) of the original circle. It follows that the image w^* of z under the analytic function S_δ is situated on the unscaled infinitesimal circle around $S_\delta(a) = \overline{a}$ but it is rotated around \overline{a} by -2α compared to z relative to a . Here, α is the angle between the tangent to $\delta(t)$ at a and the x -axis. Finally, point w^* is conjugated along the x -axis and we conclude that $R_\delta(z) = \overline{w^*}$. See Figure 3.10 for a visualization.

Proposition 3.3.1 *The generalized reflection $R_\delta(z) = \overline{S_\delta(z)}$ on an analytic curve δ is locally the reflection of z across the tangent at a curve point a .*

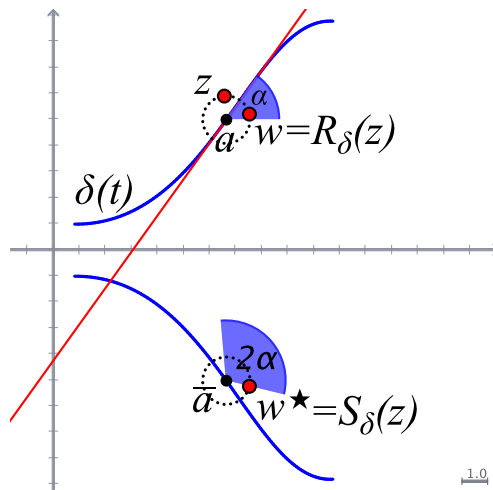


Figure 3.10: Analytic reflection along curve δ : $R_\delta(z) = w$.

This knowledge about the extension of a conformal map along analytic curves will be important later in Chapter 5.2.2, where we actually compute the conformal approximation of the preimage strip to the stroke.

It is also important to note that the local reflection at the tangent of the analytic curve is unique only for points z in an infinitesimal neighborhood of curve points where the tangent is uniquely defined. Therefore, if there is a non-regular point on the envelope, this method cannot be applied within the infinitesimal neighborhood of the point of regression.

Furthermore, we have assumed that the stroke is not self-intersecting. We will address this case in Section 5.4.1.

From Definition 2.2.5, we know that the boundary of the stroke also contains two circular arcs from $\mathcal{C}(0)$ and $\mathcal{C}(T)$. For points near these circular arcs, the classical Schwarz reflection principle can be applied, and circular reflection at $\mathcal{C}(0)$ and $\mathcal{C}(T)$ provides the correct solution.

Additionally, we recall from Chapter 2 that strokes are defined as the union of all circles $\mathcal{C}(t)$ along the curve γ with radius $r(t)$ for $t \in I = [0, T]$. For regular boundary points, Equation (2.20) states that the limiting, discriminant, and tangential envelopes coincide. Therefore, the boundary of the stroke is tangent to a circle $\mathcal{C}(t)$ at all regular points a . The generalized reflection along the boundary is equal to the reflection at the tangent at a . Locally, the tangent is a component of the envelope by the definition of the tangential envelope E_t , and it is tangent to $\mathcal{C}(t)$ for $a = (a_1, a_2)$ and $F(t, a_1, a_2) = 0$. Therefore, it is also possible to consider reflecting z on the circle $\mathcal{C}(t)$ for a point z infinitesimally close to the regular boundary point a .

Note 3.3.2 *Despite the relationship between tangents, circles, and regular curves, it is important not to confuse the possible reflection along circles $\mathcal{C}(t)$ of the stroke with the reflection at circles of curvature of the envelope curves. Even though the envelope curve δ is tangent to both circles, namely the circle $\mathcal{C}(t)$ and the circle of curvature of δ , they are generally not identical. Figure 3.11 shows an example. The simplest argument to see that the circle of curvature and $\mathcal{C}(t)$ do not coincide in general is that for constant radius function $r(t) \equiv r_c$ the boundary would have to be a circle of constant curvature $\kappa = \frac{1}{r_c}$.*

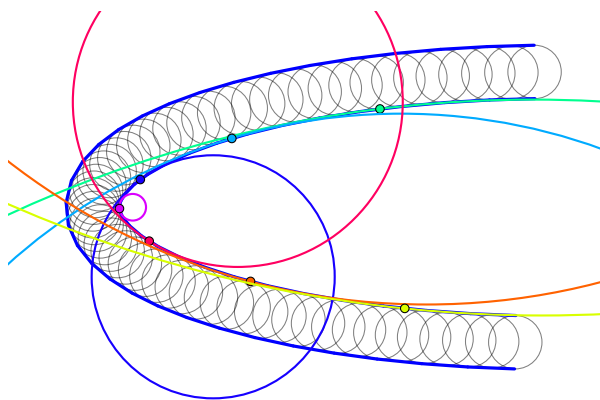


Figure 3.11: Circles of curvature for the inner envelope curve of an elliptic stroke.

4 | Discrete Data and Conformality

So far, Chapters 2 and 3 have focused on the stroke model and conformality within continuous settings. We have defined the stroke by a continuous curve γ and a continuous radius function r , and the strip and stroke domains as subsets of the complex plane. This, however, is only an idealized setting.

As emphasized earlier, the stroke model is based on user-drawn input on digital devices. The user-drawn input is discretely registered by the computer and used to generate a radius function, based on the pressure applied by the user on the drawing surface. Thus, a continuous model does not accurately represent the given circumstances. Additionally, the tiled strip is composed of image textures T , and also the stroke is a subdomain of an image texture. A texture is a digital image consisting of a set of discrete data points, known as pixels. These pixels are arranged in a rectangular grid. The number of pixels in a texture is a measure of the resolution of the grid. Thus, the preimage and target domains of the conformal maps, namely the rectangular strip and the stroke, are subsets of discrete pixel grids rather than subsets of the continuous complex plane.

This requires the investigation of appropriate discretizations of the continuous stroke model and the notion of conformality. In this context, appropriate means that the characteristic properties of the stroke have to be represented in the discrete setting and that the discrete understanding of conformality serves the given conditions, e.g., the definition of a texture as a discrete pixel grid.

In Section 4.1, we will examine the required modifications to the continuous stroke model caused by discrete input data. In Sections 4.1.1 and 4.1.2, we will compare different spline interpolation methods for curve γ and radius function r . We will decide for cubic B-spline interpolation to model our strokes because it ensures that the interpolated curve γ and radius function r are C^2 -continuous, and the envelope curves w_{\pm} are C^1 -continuous. For the discrete definition of the stroke on the basis of the interpolated curve γ and r we will present an algorithm in Section 4.1.3 that selects equidistant points C_{t_d} on γ , which act as circle centers for the stroke defining circles $\mathcal{C}(t_0)$ to $\mathcal{C}(t_N)$.

Apart from discrete input data, we will also investigate discrete conformality. In Section 4.2, we will examine two conceptions of discrete conformality that are capable of reflecting relevant properties of continuous conformality when applied to discrete domains: circle packings and discrete conformal equivalence. The theory of discrete conformal equivalence, which is based on metric scaling and length-cross-ratios, will serve as the basis for the evaluation of our algorithm for conformality in Section 5.3.1. The discrete conformal theory of circle packings introduces the concept of local univalence, acting as a model for our self-intersecting strokes. The notion of branch circles will serve as

an inspiration for the treatment of multiply covered regions of our strokes due to self-intersection or singular boundary points (see Section 4.3).

In order to deal with self-intersecting strokes and regions with singular boundary points in the algorithm described in Chapter 5, we will present techniques for detecting these special cases in Section 4.3. We will provide a criterion for new circles of the stroke based on their intersections with existing circles that indicates that the stroke is self-intersecting. Additionally, we will introduce a discrete technique to identify singular boundary points and other significant stroke points related to the swallowtail surface associated with the stroke known from Section 2.6.2. In cases of singular boundary points, we will classify several multiply covered regions of the stroke by explicitly determining their boundary curves.

4.1 Interpolation of curve γ and radius function r

The objective of interpolation is to refine the set of discrete data points registered by the computer or tablet on the curve drawn by the user. The registered data points P_{t_d} for $t_d \in I$ can either be very close together or quite far apart, depending on the sampling rate of the electronic device and the speed of the user's drawing. In any case, they are irregularly distributed over the interval $I = [0, T]$, which is equal to $[t_0, t_N]$ for $N + 1$ registered data points. By redistributing, removing, or adding data points, we aim to achieve a sufficient number of data points and thus circles to define a stroke. Furthermore, we attempt to accurately capture the user's input by avoiding significant changes to the drawn path and preserving the smoothness of the drawing.

There exist several methods for point interpolation. In the following, we will study three different types of spline interpolation and examine the advantages and disadvantages of these methods for our purpose.

Splines are piecewise polynomials that are used to interpolate data points locally. The term "splines" originated from a mechanical tool that craftsmen utilized to interpolate smooth curves through given points, such as design elements of ships or railroad tracks [Far02, Chapter 14.1].

The spline polynomials have a specific degree n , and they are defined separately in subintervals between discrete data points. In addition, splines have a certain degree of smoothness, typically at least continuous, but may also be C^1 - or C^2 -continuous, which ensures that the individual spline pieces not only meet, but also have the same tangent or even the same curvature at the connecting data points [SM03, Chapter 11.1].

Note that the symbols γ and r will be used henceforth for the piecewise interpolated curve or radius function composed of individual spline segments.

4.1.1 Linear interpolation and cubic Hermite splines

The linear spline is the simplest form of spline. It connects each data point P_{t_d} to its neighbor $P_{t_{d+1}}$ with a line segment $s_d(t) = (1 - t)P_{t_d} + tP_{t_{d+1}}$ for $t \in [0, 1]$. This results in a spline that is C^0 -continuous, meaning it is connected at the discrete data points, but lacking higher degrees of smoothness. The benefits of linear interpolation include the easy calculation of these splines and the fact that the resulting curve interpolates the data

points. The individual line segments s_d are computed locally, since each data point P_{t_d} only affects the two neighboring splines s_{d-1} and s_d . A new line segment s_d may be added to the interpolated, piecewise linear curve γ without difficulty.

Linear interpolation can also be applied to radius values obtained from registered pressure data. Consequently, the interpolated radius function r would consist of a piecewise linear C^0 -continuous splines, i.e., of linear functions $r_d(t) = (1-t)r_{t_d} + t \cdot r_{t_{d+1}}$.

As modern computers possess sufficient computing power, the discrete sample points tend to be close together. This results in short linear segments and a reasonable approximation of the curve γ and radius function r . Nevertheless, if the computer lacks computing power or the user draws very fast and the sample points are too far apart, the linear segments are visible in the resulting stroke, as shown in Figure 4.1.



Figure 4.1: Linear spline interpolation: linear segments visible in the stroke.

When the curve γ and radius function r are provided via C^0 -continuous interpolation, the envelope curves can be calculated explicitly by Equation (2.13). Since γ and r are piecewise defined C^0 -continuous splines, the envelope is defined by s_d and r_d along with their respective derivatives s'_d and r'_d on the corresponding intervals between data points:

$$w_{\pm}^d(t) = s_d(t) - \frac{r_d(t)r'_d(t)}{\|s'_d(t)\|^2} s'_d(t) \pm \frac{r_d(t)}{\|s'_d(t)\|^2} \sqrt{\|s'_d(t)\|^2 - r'_d(t)^2} (s'_d(t))^{\perp} \quad (4.1)$$

The disadvantage of linear interpolation is that the derivatives of the interpolated curve and radius functions are constant on the intervals between the discrete data points, namely $s'_d(t) = P_{t_{d+1}} - P_{t_d}$ and $r'_d(t) = r_{t_{d+1}} - r_{t_d}$. This implies that the derivatives of s_d and r_d do not coincide at the common endpoints, as C^0 -continuity already suggested. As a consequence, the piecewise defined envelope curves as defined in Equation (4.1) are not connected and, therefore, not even continuous.

This is why we examine other spline interpolation methods to make the interpolated curve γ and radius function r smoother and the envelope curves at least continuous.

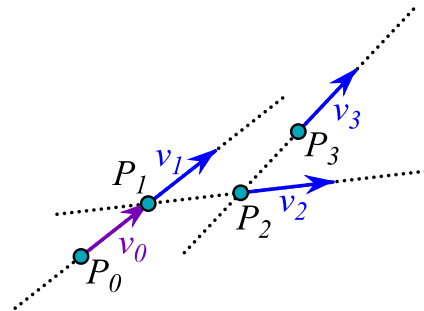
The cubic Hermite spline is another well-known type of spline. ‘‘Cubic’’ indicates that each pair of consecutive data points P_{t_d} and $P_{t_{d+1}}$ is connected by a cubic function $h_d(t) = at^3 + bt^2 + ct + d$ for real constants $a, b, c, d \in \mathbb{R}$ not all equal to zero. The functions h_d are defined by discrete data points P_{t_d} and tangent vectors v_{t_d} which are fixed at those data points. Thus, h_d and h_{d+1} share the point $P_{t_{d+1}}$ and the tangent $v_{t_{d+1}}$.

The explicit formula for h_d between P_{t_d} and $P_{t_{d+1}}$ is given by [Far02, Chapter 14.4.1]

$$h_d(t) = \begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} P_{t_d} \\ v_{t_d} \\ P_{t_{d+1}} \\ v_{t_{d+1}} \end{pmatrix}. \quad (4.2)$$

As we can see, the data point P_{t_d} is met by $h_d(0)$ and the derivative of the spline at zero gives $h'_d(0) = v_{t_d}$. Moreover, it holds that $h_d(1) = P_{t_{d+1}}$ and $h'_d(1) = v_{t_{d+1}}$. This ensures that the entire spline, which is piecewise given by Equation (4.2), is C^1 -continuous and interpolates all of the given discrete data points. Furthermore, Hermite splines offer potentially attractive interpolation for curve γ and radius function r as each individual spline function only relies on the data point and tangent at its two end points. Hence, interpolation is done locally. Additionally, the matrix format in Equation (4.2) allows for quick and efficient computations.

However, it remains uncertain which tangents should be used in our setting. When a new point $P_{t_{d+1}}$ is added to the set of registered discrete data points, we don't know in which direction the user will continue to draw the stroke. We must therefore estimate the direction of tangent $v_{t_{d+1}}$ at this new data point. We use the vector $v_{t_{d+1}} = P_{t_{d+1}} - P_{t_d}$ to represent the missing tangent vector. The missing tangent at P_0 is set to $v_0 = P_1 - P_0$.



The same Equation (4.2) is used to interpolate the discrete data points for the radius function r . If both C^1 -continuous interpolated splines h_d and r_d of curve γ and radius function r are given, the piecewise defined envelope curves of the stroke can be calculated as in Equation (4.1) replacing s_d by h_d . Since the interpolation properties ensure that $h_d(1) = h_{d+1}(0)$, $r_d(1) = r_{d+1}(0)$, $h'_d(1) = h'_{d+1}(0)$, and $r'_d(1) = r'_{d+1}(0)$, the envelope curves satisfy $w_d(1) = w_{d+1}(0)$. This is an improvement over the envelope of the linearly interpolated curves. However, C^0 -continuity is the best that can be achieved. Consequently, the envelope curves do not share the same tangents at joint curve points, as illustrated by the red envelope of the stroke in Figure 4.2.

4.1.2 Cubic B-splines

To achieve smooth envelope curves, we examine cubic B-splines. B-splines have advantages over linear or Hermite splines, given that they are C^2 -continuous, resulting in smoother envelope curves that are C^1 -continuous. However, B-splines have two critical drawbacks: there is a partial loss of locality in the computations, and the given discrete data points are not interpolated by the curve γ . Therefore, we will now analyze this option in detail.

The “B” in B-splines stands for “basis” [Kno99, Section 3.3, p.151], indicating that basis functions are combined to obtain the interpolated cubic curve. Cubic basis functions, denoted as B_d , are defined for the given data points P_{t_d} , where $t_d \in [t_0, \dots, t_N]$. The basis

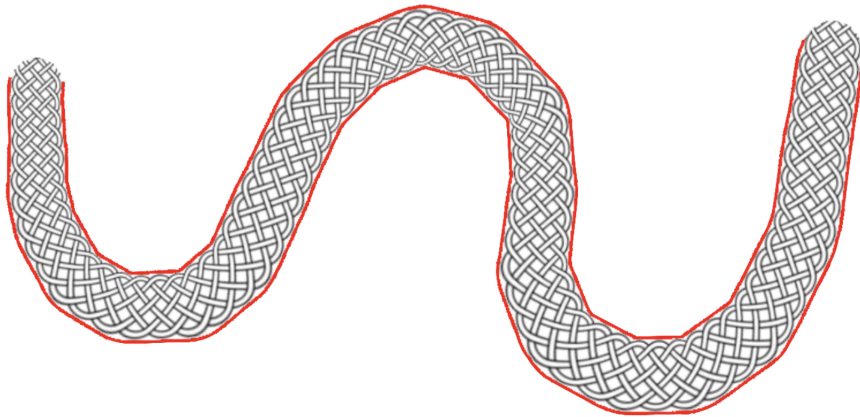


Figure 4.2: Hermite spline interpolated stroke with C^0 -continuous envelope.

functions regulate the influence of each data point P_{t_d} on the resulting spline over each interval $[t_d, t_{d+1}]$ [Far02, Chapter 15.3].

The basis functions B_d must satisfy specific criteria to ensure C^2 -continuity of the cubic spline [Far02, p.354], [Kno99, p.153f.]. These criteria are the following: the basis functions must be positive at all times and add up to one, ensuring that the interpolated curve lies within the convex hull of the data points P_{t_d} . Furthermore, each basis function $B_d(t)$ should have compact support and be equal to zero for $t \notin [t_{d-1}, t_{d+2}]$. This results in point P_{t_d} influencing four consecutive spline pieces. Additionally, the basis functions must be C^2 -continuous to ensure C^2 -continuity of the resulting spline.

According to Farouki [Far02, p.355], all basis functions are identical except for the translation into the respective support interval. Thus, for $t \in [t_i, t_{i+1}]$, $B_i(t)$ equals $B_{i-1}(\tilde{t})$ with $\tilde{t} \in [t_{i-1}, t_i]$. As a result, we can examine the basis functions B_0 , B_1 , B_2 , and B_3 for the spline $b_1(t)$ in the interval $t \in [t_1, t_2]$ defined by

$$b_1(t) = B_0(t) \cdot P_0 + B_1(t) \cdot P_1 + B_2(t) \cdot P_2 + B_3(t) \cdot P_3$$

and apply the findings to all other basis functions.

We assume, without loss of generality, that all intervals are normalized to $t \in [0, 1]$. To ensure C^2 -continuity of the cubic spline, the following equations must be satisfied. They guarantee that the basis functions B_0 , B_1 , B_2 , and B_3 satisfy the above requirements.

To get C^2 -continuity at the limits of the support of each basis function, it must hold [Far02, p.356]

$$\begin{aligned} B_0(1) &= 0, & B'_0(1) &= 0, & B''_0(1) &= 0, \\ B_3(0) &= 0, & B'_3(0) &= 0, & B''_3(0) &= 0. \end{aligned} \quad (4.3)$$

To ensure C^2 -continuity of the basis functions within the support, and taking into account that the basis functions B_i are shifted copies of each other, it must also be satisfied that

$$\begin{aligned} \text{for } C^0\text{-continuity:} & \quad B_0(0) = B_1(1) & B_1(0) = B_2(1) & B_2(0) = B_3(1) \\ \text{for } C^1\text{-continuity:} & \quad B'_0(0) = B'_1(1) & B'_1(0) = B'_2(1) & B'_2(0) = B'_3(1) \\ \text{for } C^2\text{-continuity:} & \quad B''_0(0) = B''_1(1) & B''_1(0) = B''_2(1) & B''_2(0) = B''_3(1) \end{aligned} \quad (4.4)$$

Also, it is required that the basis functions sum to one at all times:

$$B_0(t) + B_1(t) + B_2(t) + B_3(t) = 1. \quad (4.5)$$

We aim to express the cubic B-spline b_1 using P_0, \dots, P_3 in a manner similar to the Hermite spline as in Equation (4.2) by

$$b_1(t) = \begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix} \underbrace{\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{pmatrix}}_{=:C} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}.$$

The 16 equations in (4.3), (4.4), and (4.5) uniquely define the 16 entries c_0, \dots, c_{15} of the matrix C because the following equations must be satisfied for $t \in [0, 1]$:

$$\begin{aligned} B_0(t) &= c_0 + c_4t + c_8t^2 + c_{12}t^3 \\ B_1(t) &= c_1 + c_5t + c_9t^2 + c_{13}t^3 \\ B_2(t) &= c_2 + c_6t + c_{10}t^2 + c_{14}t^3 \\ B_3(t) &= c_3 + c_7t + c_{11}t^2 + c_{15}t^3 \end{aligned}$$

As derived by Holmér [Hol22, 51:52 – 53:36], this system of equations solves to

$$b_1(t) = \begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix} \frac{1}{6} \begin{pmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}. \quad (4.6)$$

Using Formula (4.6), it is simple to input the provided data points $P_{t_{d-1}}, P_{t_d}, P_{t_{d+1}}$, and $P_{t_{d+2}}$ in order to obtain the interpolating B-spline b_d within the interval $[t_d, t_{d+1}]$. However, there are drawbacks to this C^2 -continuous interpolation. Primarily, the computations are not entirely local, as the spline b_d depends not only on the data points P_{t_d} and $P_{t_{d+1}}$, which bound the interval for which b_d is defined, but also on the two data points P_{t_d} and $P_{t_{d+1}}$. Consequently, there is a lack of data points to calculate B-splines for the initial and last interval between data points P_0 and P_1 as well as between $P_{t_{N-1}}$ and P_{t_N} if P_{t_N} is the last data point registered on the user-drawn curve.

We address this problem by artificially adding a point P_{-1} as the mirror image of P_1 at P_0 : $P_{-1} = P_0 + (P_0 - P_1)$. However, we do not add a data point P_{N+1} for the last interval in order to avoid inconsistencies in the continuity properties at the boundary between the cubic spline pieces and to avoid having to change already calculated stroke parts when the actual data point $P_{t_{d+1}}$ is included in the model. If it is assumed that modern computers possess fast enough data recognition so that the registered data points are very close together, i.e., they only have a distance of very few pixels, it is no problem to postpone the calculation of the spline for the last interval until a new point is given. Consequently, the last spline for N registered data points is the one defined by $P_{N-3}, P_{N-2}, P_{N-1}$, and P_{t_N} (see Figure 4.3 at the right end of the stroke).

The second issue is that the splines b_d do not interpolate the provided data points P_{t_d} , but instead, they modify the trajectory of the user-drawn curve between the registered

data points. However, the deviation is negligible given the large number of closely spaced data points. Hence, modifying the originally drawn curve can be accepted, as it leads to better boundary properties of the stroke. Figure 4.3 shows how close the red data points are to the blue interpolated curve.

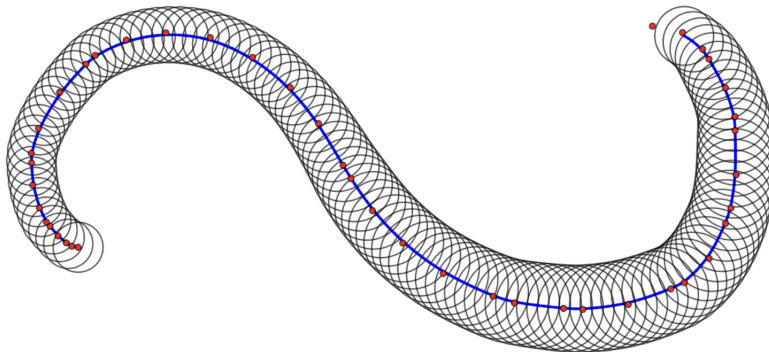


Figure 4.3: Issues of cubic B-spline interpolation.

To summarize: we have a piecewise defined curve γ , which is C^2 -continuous given by

$$\gamma(t) := \begin{cases} b_0 \left(\frac{(t-t_0)}{(t_1-t_0)} \right) = b_0 \left(\frac{t}{t_1} \right) \\ \vdots \\ b_d \left(\frac{(t-t_d)}{(t_{d+1}-t_d)} \right) \\ \vdots \\ b_{N-1} \left(\frac{(t-t_{N-1})}{(t_N-t_{N-1})} \right) = b_{N-1} \left(\frac{(t-t_{N-1})}{(T-t_{N-1})} \right) \end{cases} \quad \text{for } t \in [0, T] = [t_0, t_N]. \quad (4.7)$$

The radius function r is defined piecewise by splines r_d as in Equation (4.7) using the same cubic B-spline interpolation. This allows us to examine the properties of the associated envelope curves w_{\pm}^d as in Equation (4.1). The use of B-spline interpolation ensures C^1 -continuity of the envelope curves since the splines b_d and r_d , along with their derivatives and second derivatives, coincide for the transitions between intervals (see Equations (4.3) and (4.4)). Figure 4.4 shows a stroke interpolated using cubic B-spline. Comparing the C^1 -continuous envelope of this stroke with the C^0 -continuous envelope of the stroke interpolated using Hermite spline in Figure 4.2 highlights the improvement.

Furthermore, the envelope curves w_{\pm}^d are analytic because both γ and r are cubic polynomials, resulting in their derivatives also being polynomials and, therefore, analytic [Nee23, p.226]. Condition (2.2) guarantees that $\|\gamma'(t)\|^2 - r'(t)^2 > 0$, which means that the real square root of this function is also analytic. Therefore, Equation (4.1) obtains the envelope curves as a combination of analytic functions through addition, subtraction, multiplication, and division which makes the envelope curves analytic [Nee23, p.228].

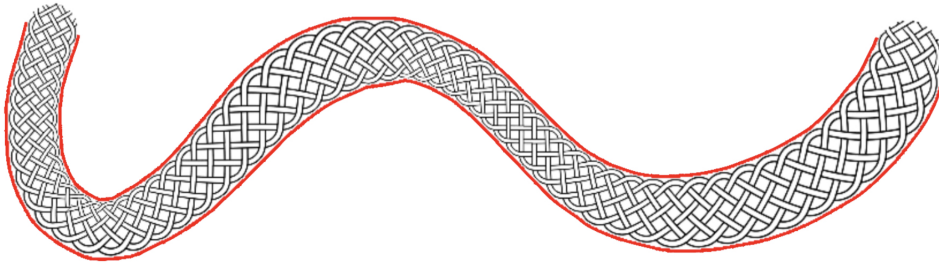


Figure 4.4: Cubic B-spline interpolated stroke with C^1 -continuous envelope curves.

Proposition 4.1.1 (Analytic envelope curves)

If the curve γ and radius function r are interpolated using cubic B-splines from the registered data points and pressure input, the resulting envelope curves w_{\pm}^d are analytic.

This allows us to apply the results of Section 3.3 on the generalization of the Schwarz reflection principle to the envelope curves interpolated in this way. Overall, the benefits outweigh the drawbacks, which can be neglected for sufficiently fast computers. Therefore, we will use cubic B-spline interpolation for our discrete stroke model.

4.1.3 Discrete equidistant points on the interpolated curve γ

From the discrete data points P_{t_d} that are registered on the user-drawn path, a curve γ and a radius function r , both C^2 -continuous, are piecewise interpolated using cubic B-splines b_d and r_d , as it was examined in the last section (see Equations (4.6) and (4.7)). When a curve γ and a function r are given, the stroke is uniquely defined by the associated continuous family of circles. But this continuous definition is not used to actually draw the stroke. Instead, a discrete set of closely spaced circles is drawn, similar to previous figures depicting families of circles, such as those found in the Benchmark Examples 2.1.6 to 2.1.10. Therefore, it is essential to choose the centers for the discrete set of circles on the interpolated curve γ appropriately. If the centers are not sufficiently close, the circles can become disconnected and create a broken stroke or cause a visibly wavy envelope of the circles (see Figure 4.5).

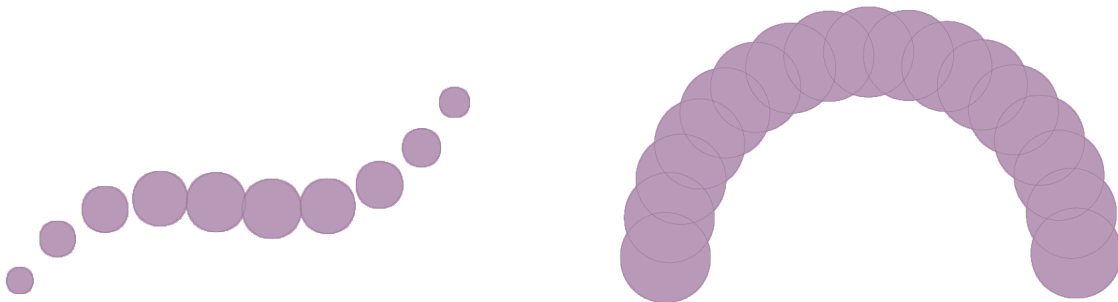


Figure 4.5: Placing circles too far apart causes disjoint or wavy strokes.

If the centers of the circles are located too close together, the simulation may become slow due to many calculation steps, or the circles may violate the “supersonic” Condition (2.2), which prevents circles from being nested. To exclude centers that are too close or too far apart, a new circle is placed along the interpolated curve γ every few pixels, depending on the resolution of the texture on which the stroke is drawn.

A new circle center on curve γ is chosen according to the following procedure. The corresponding pseudocode is stated in Algorithm 1. Let $0 < \epsilon < 1$ and $0 < \delta < 1$ be small constants, and let p denote the pixel distance between the centers of two consecutive circles. The value of ϵ represents the permitted deviation from the pixel distance p between circle centers. Constant δ is the step width of the search for new centers on γ . Let C_{t_d} be the previously selected circle center on γ at $\gamma(t_d)$, and let r be the interpolated radius function. If $|\|\gamma(t_d + \delta) - C_{t_d}\| - p| \leq \epsilon$ and if $r'(t_d + \delta)^2 < \|\gamma'(t_d + \delta)\|^2$, then a new circle is created around $C_{t_{d+1}} = \gamma(t_d + \delta)$ with radius $r_{t_{d+1}} = r(t_d + \delta)$. Otherwise, the next curve point $\gamma(t_d + 2\delta)$ is examined.

Algorithm 1 Finding an equidistant circle along the interpolated curve γ .

```

 $\epsilon > 0, \delta > 0$ 
 $\delta^* \leftarrow \delta$ 
 $p \leftarrow$  small amount of pixels
 $C_{t_d} \leftarrow \gamma(t_d)$ 
 $continue \leftarrow true$ 
while  $continue$  do
  if  $|\|\gamma(t_d + \delta^*) - C_{t_d}\| - p| \leq \epsilon$  and  $r'(t_d + \delta^*)^2 < \|\gamma'(t_d + \delta^*)\|^2$  then
     $C_{t_{d+1}} \leftarrow \gamma(t_d + \delta^*)$ 
     $r_{t_{d+1}} \leftarrow r(t_d + \delta^*)$ 
     $continue \leftarrow false$ 
  else
     $\delta^* \leftarrow \delta^* + \delta$ 
  end if
end while

```

Our tests indicate that using a pixel distance of $p \in (2, 5)$ and parameters $\epsilon \in (0.2, 0.5)$ and $\delta = 0.01$ is a suitable choice for a texture consisting of 1024×512 pixels.

The so defined set of equally spaced curve points C_{t_d} lies on the interpolated piecewise cubic spline curve γ . The registered data points P_{t_d} are not necessarily a subset of the points C_{t_d} . The resulting stroke consists of a discrete set of circles $\mathcal{C}(t_d)$ belonging to the family of circles defined by γ and the likewise interpolated radius function r . A so resulting stroke is, for example, depicted in Figure 4.3.

4.2 Discrete conformality

It was noted in the literature review in Section 1.3 and in Chapter 3 that computing conformal maps between simply connected planar domains is a challenging task, even when their existence is guaranteed. Beyond using numerical methods to approximate continuous maps, however, there are several options for discrete analogues. They can serve as a starting point for refinement toward a continuous result, or as standalone methods for discrete settings. In a discrete setting, the considered domains are polyhedral surfaces, i.e., they consist of polygons glued edge-to-edge [BSS16, p.57], [BPS15, p.4]. Typically, these polygons are triangles or quadrilaterals, and the resulting surfaces are referred to as triangulations or quadrangulations.

Since a stroke is defined as a subset of a discrete texture consisting of a finite number of pixels forming a rectangular grid, the stroke domain can be viewed as a polygonal surface consisting of quadrilaterals or even triangles depending on the considered “edges” between pixels (see Section 5.3.1). A pixel grid in the stroke of Benchmark Example 2.1.6 is illustrated in Figure 4.6 on the left. The texture resolution has a direct impact on the grid, where higher resolutions lead to finer grids and lower resolutions result in coarser grids. Figure 4.6 on the right shows the quadrangulation obtained from the pixel grid by adding horizontal and vertical edges between adjacent pixels.

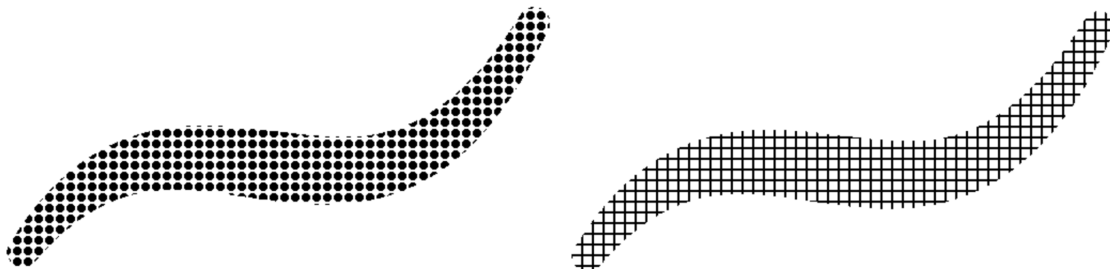


Figure 4.6: The pixel grid underlying the stroke and the related quadrangulation.

Taking into consideration that our strokes are discrete domains, we will investigate different definitions of discrete conformality and their characteristics. Also, we will explore how to demonstrate the conformality of a computed map using discrete local characteristics, without knowledge of the continuous counterpart of the map or the conformal modulus of the rectangle in its conformal equivalence class.

To define discrete conformality, the question is which of the equivalent notions of conformality in the continuous case are appropriate for a discrete counterpart. Crane gives a good overview [Cra19]:

Most of the continuous notions of conformal maps F lead to analogues of discrete conformal maps f for triangulated surfaces, which are too rigid to be used. If the angles of a triangulation are preserved, this restricts the discrete conformal map f to be a similarity transformation with uniform scaling of the edge lengths. If the continuous conformal map F is defined as $F = u + iv$ for functions u, v satisfying the Cauchy-Riemann equations $u_x = v_y$ and $u_y = -v_x$, this implies that the discrete analog f is a global isometry up to uniform dilation. The discrete conformal map f can also be found as the critical

point of a discrete analog of the Dirichlet energy, which, however, is related to the solution of the discretized Cauchy-Riemann equations, and is thus subject to the same rigidity. Another continuous concept that has been found to be too rigid is the Hodge star duality.

If the continuous map F is defined as the sum $F = a+ib$ of a pair of conjugate harmonic functions a and b , this provides a description of the discrete conformal function f , which is more flexible. In this case, f is related to the discrete Dirichlet energy, and harmonic functions are well understood in the discrete case. However, this particular approach is one of the finite element methods that relate well to the smooth setting in their limit of refinement. As standalone method for a fixed triangulation, the desired properties of continuous conformality are usually not met. Additionally, since we compute our stroke and corresponding map for a fixed texture resolution with a constant number of pixels, the refinement arguments are interesting for choosing the texture resolution, but they are not of much use once the textures are defined.

Chen and Gotsman compute a conformal map f from a regular polygonal mesh inside a polygon P to a target polygon Q as a minimizer of a conformal energy function [CG17]. To obtain f , barycentric coordinates for the points in the polygons are calculated. To achieve this, basis functions B_j are defined for each preimage vertex z_j on the boundary of P so that every point $z \in P$ can be expressed as $z = \sum_j z_j \cdot B_j(z)$. The image of z , $w = f(z)$, is then written as a linear combination with the same basis function coefficients $B_j(z)$, but based on the target vertices w_j on the boundary of Q : $w = f(z) = \sum_j w_j \cdot B_j(z)$. These basis functions B_j are used to compute a conformal energy function for each polygon in the domain. Minimization of this energy function produces a conformal map f if and only if the conformal energy is zero. It is shown that harmonic functions give good solutions to this minimization problem. The conformal energy is evaluated separately for each polygon in the mesh and added up to determine the energy of the entire domain. It is possible to weight individual energy summands in such a way that more important regions of the polygonal domains have more influence on the conformal energy than others. By prioritizing regions with higher weight, they can achieve better approximations of zero conformal energy, while the other regions yield poorer results. This approach suggests that an (approximately) conformal map is achieved when the energy function is (almost) zero. Furthermore, the energy is calculated for each polygon in the polygonal domain individually and then combined.

We will adapt this method in Sections 5.3.1 and 5.3.2, where we evaluate our algorithm for computing a conformal map. Our procedure involves verifying that particular constraints are met for every pixel within the stroke domain, calculating the sum of all results, and minimizing the overall error.

This leads us to the question of how to calculate a discrete conformal map and what its characteristics are. To address this, we examine the two remaining notions of conformality mentioned by Crane [Cra19]: metric scaling and circle preservation. Both produce good results for discrete conformal maps on polygonal surfaces, both in a discrete setting and in the limit of mesh refinement, which relates them to the continuous case. These two notions of circle preservation and metric scaling open up the research areas of circle packings and discrete conformal equivalence, respectively, which we will discuss in the next two sections.

4.2.1 Discrete conformal equivalence of polyhedral surfaces

In the continuous context, two Riemann metrics, g and \tilde{g} on a smooth manifold M are called conformally equivalent if, for some conformal factor $u \in C^\infty(M)$, they are related by metric scaling:

$$\tilde{g} = e^{2u}g.$$

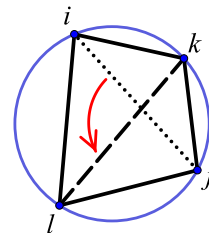
Thus, there exists an equivalence class of metrics on M that are connected by a conformal factor and that are all conformally equivalent to each other. To discretize this concept of conformality, the smooth manifold M is substituted with a piecewise Euclidean manifold S , which is connected and triangulated by $T = (V, E, F)$ with a set of vertices $i \in V$, edges $ij \in E$ and triangular faces $ijk \in F$. The Riemann metrics g and \tilde{g} are substituted with the discrete edge length metrics $\ell, \tilde{\ell}$. The metric $\ell: E \rightarrow \mathbb{R}^+$ allocates positive lengths ℓ_{ij} to the edges $ij \in E$, which satisfy the triangle inequalities for all faces $ijk \in F$: $\ell_{ij} + \ell_{jk} > \ell_{ki}$. [Cra19, BPS15, BSS16, SWGL15, GLSW18, LSW22]

There are two discrete settings that need to be distinguished. In one setting, discrete conformality is defined on a manifold S between two triangulations T and \tilde{T} that are combinatorially equivalent. In the other setting, these two triangulations are combinatorially different.

If the triangulations of S are different, then the focus of the investigation shifts to the manifold itself. Discrete conformality between two combinatorially different triangulations of a surface S is defined on the basis of intrinsic Delaunay triangulations and edge flips. In the work of Gu et al. [SWGL15, GLSW18], two combinatorially different triangulations T and \tilde{T} are said to be discretely conformally equivalent if there is a sequence of triangulations $T = T_0, T_1, T_2, \dots, T_m = \tilde{T}$ of S and a sequence of metrics $\ell = \ell^0, \ell^1, \ell^2, \dots, \ell^m = \tilde{\ell}$ such that three conditions hold:

- 1) All triangulations T_d are Delaunay with respect to the corresponding metric ℓ^d . This means that for every pair of triangles $ijk, jil \in F$, the sum of opposite angles α_k^{ij} and α_i^{ji} with respect to the common edge ij is less than π .
- 2) If the triangulations T_d and T_{d+1} are equal in two consecutive steps, the metrics ℓ^d and ℓ^{d+1} differ and are related by metric scaling using a discrete conformal factor $w: V \rightarrow \mathbb{R}$ that assigns constant factors to all vertices of the triangulation:
 $\ell_{ij}^{d+1} = e^{(w_i + w_j)} \cdot \ell_{ij}^d.$

- 3) If two consecutive metrics ℓ^d and ℓ^{d+1} are equal, the triangulations T_d and T_{d+1} differ by a cocircular edge flip, i.e., the common edge ij between two cocircular triangles ijk and jil is replaced by the edge kl between the former opposite vertices as shown in the inset.



Based on this definition of discrete conformality between combinatorially different triangulations, a discrete uniformization theorem can be derived using the notion of cone angles. For each vertex $v \in V$ of a triangulation, the cone angle $K(v)$ for interior vertices v is defined as the difference between 2π and the sum of all angles around v . For boundary

vertices v , $K(v)$ equals the difference between π and the sum of all angles around v . The total sum of all cone angles $K(v)$ satisfies the Gauss-Bonnet condition $\sum_{v \in V} K(v) = 2\pi \cdot \chi(S)$ where $\chi(S) = |V| - |E| + |F|$ is the Euler characteristic of surface S .

The discrete uniformization theorem states that for a closed connected surface S with a discrete metric ℓ and prescribed cone angles K , there exists a unique metric $\tilde{\ell}$ obtained from ℓ via metric scaling that satisfies the prescribed cone angles K [SWGL15, Theorem 1.2]. This theorem guarantees existence and uniqueness for discrete conformal equivalent metrics of a given surface with prescribed cone angles.

Also for the discrete case, existence and uniqueness statements are not sufficient to find concrete discrete conformal equivalent metrics. However, the sequence of discrete metrics ℓ^0, \dots, ℓ^m and Delaunay triangulations T_0, \dots, T_m in the above notion of discrete conformality is an evolution of conformal factors w over time. Considering this evolution yields a convex energy function and since it is convex, a minimizing solution w^* can be found in a finite amount of time if it exists. For the specific case of a Jordan domain with three boundary points, a convergence theorem to the continuous Riemann map can be proven. [Cra19, LSW22]

Note that the sequences of triangulations and metrics that satisfy conditions 1 to 3 result in a flow from one triangulation to the next, ensuring discrete conformal equivalence. As stated in the introduction, our algorithm aims to achieve visible convergence towards a conformal solution. Hence, the technique of gradually altering local properties of a polygonal grid will be revisited in Chapter 5 where our algorithm is presented.

Metric scaling $\ell_{ij} = e^{(w_i+w_j)} \cdot \ell_{ij}$ with a discrete conformal factor w is also part of the notion of discrete conformality between two combinatorially equivalent triangulations T and \tilde{T} of a surface S . This case has been widely studied by Bobenko et al. [BPS15, BSS16]. There again, the discrete metrics ℓ and $\tilde{\ell}$ assign positive lengths to the edges of the triangulations T and \tilde{T} , respectively, satisfying the triangle inequalities. Discrete conformal equivalence between T and \tilde{T} is defined as follows.

Definition 4.2.1 (Discrete conformal equivalence) [BPS15, Def. 2.1.1]

Two combinatorially equivalent Euclidean triangulations (T, ℓ) and $(\tilde{T}, \tilde{\ell})$ are discretely conformally equivalent if the discrete metrics ℓ and $\tilde{\ell}$ are related by

$$\tilde{\ell}_{ij} = e^{\frac{1}{2}(w_i+w_j)} \ell_{ij} \quad (4.8)$$

along every edge ij between two vertices i and j for some conformal factor $w \in \mathbb{R}^{|V|}$.

This defines an equivalence relation on the set of discrete metrics on combinatorially equivalent triangulations of T . An equivalence class is called a discrete conformal class of discrete metrics or a discrete conformal structure on T .

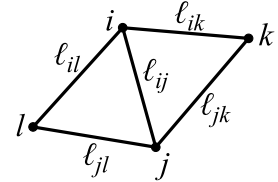
The task is to determine a conformal factor w that yields a discrete conformally equivalent metric $\tilde{\ell}$ for a given triangulation T with known edge lengths ℓ .

This could be solved by a convex energy function E which is defined on the triangulation T , the edge length values ℓ or rather the logarithmic lengths $\lambda = 2 \log(\ell)$, and the angle sum θ_i around each vertex i . The variational principles outlined in [BPS15, Chapters 3&4] and [BSS16, Chapter 3] ensure that finding a critical point of the energy

function E is equivalent to solving the discrete conformal mapping problem. As stated before, we will use the technique of minimizing a global energy function in Section 5.3.2 to find a stopping criterion for our algorithm, but not to find the conformal map itself. Instead, we consider an equivalent notion of discrete conformal equivalence for circular polyhedral surfaces.

Given a discrete metric ℓ of a triangulation $T = (V, E, F)$. Then, the length-cross-ratio based on ℓ for the interior edge $ij \in E$ between two triangles ijk and jil is defined by Bobenko et al. [BPS15, Def. 2.3.1]

$$\text{lcr}_{ij} = \frac{\ell_{il}\ell_{jk}}{\ell_{jl}\ell_{ik}}. \quad (4.9)$$



If the triangulation is embedded in the complex plane \mathbb{C} , then the length-cross-ratio lcr_{ij} equals the absolute value of the complex cross-ratio of the complex vertex coordinates z_i, z_j, z_k, z_l :

$$\text{lcr}_{ij} = \left| \frac{(z_i - z_l)(z_j - z_k)}{(z_l - z_j)(z_k - z_i)} \right|.$$

The concept of length-cross-ratios provides an additional criterion for discrete conformal equivalence among triangulations. This criterion is equivalent to Equation (4.8) in Definition 4.2.1.

Proposition 4.2.2 [BPS15, Prop. 2.3.2], [BSS16, Prop. 2.6]

Two Euclidean triangulations (T, ℓ) and $(\tilde{T}, \tilde{\ell})$ are discretely conformally equivalent if and only if the length-cross-ratios based on ℓ and $\tilde{\ell}$, respectively, are equal for each interior edge $ij \in E$: $\text{lcr}_{ij} = \text{lcr}_{ij}$.

Simply connected surfaces are bipartite if and only if their faces are even polygons, such as a quadrangulation. A polygon is *cyclic* if it is inscribed in a circle with its vertices on the circle. This results in the following statement:

Proposition 4.2.3 [BSS16, Corollary 2.10]

Two simply connected, combinatorially equivalent, Euclidean cyclic polyhedral surfaces with even faces and equipped with discrete metrics ℓ and $\tilde{\ell}$ are discretely conformally equivalent if and only if the length-multi-ratios for each face with vertices i_1, i_2, \dots, i_{2n} are equal

$$\frac{\ell_{i_1 i_2} \ell_{i_3 i_4} \cdots \ell_{i_{2n-1} i_{2n}}}{\ell_{i_2 i_3} \ell_{i_4 i_5} \cdots \ell_{i_{2n} i_1}} = \frac{\tilde{\ell}_{i_1 i_2} \tilde{\ell}_{i_3 i_4} \cdots \tilde{\ell}_{i_{2n-1} i_{2n}}}{\tilde{\ell}_{i_2 i_3} \tilde{\ell}_{i_4 i_5} \cdots \tilde{\ell}_{i_{2n} i_1}}. \quad (4.10)$$

For quadrangulations, i.e., for polyhedral surfaces whose faces $ijkl \in F$ are quadrilaterals, the equality of length-multi-ratios in Equation (4.10) guarantees that the complex cross-ratios of the vertex positions of the faces $ijkl \in F$ are equal because the absolute values of the cross-ratios equal the length-multi-ratios and the orientation of both faces is identical [BSS16, Prop. 2.12]:

$$\frac{(z_i - z_l)(z_j - z_k)}{(z_l - z_j)(z_k - z_i)} = \frac{(\tilde{z}_i - \tilde{z}_l)(\tilde{z}_j - \tilde{z}_k)}{(\tilde{z}_l - \tilde{z}_j)(\tilde{z}_k - \tilde{z}_i)}.$$

Interestingly, according to [BSS16, p.10], this theory of quadrangulations with length-cross-ratios equal to 1 is connected to circle patterns which will be examined in the following section. Also, to compare the cross-ratios, it is possible to connect the single factors $(z_j - z_i)$ for an edge $ij \in E$ through factors w_i and w_j assigned to the vertices $i, j \in V$ of the quadrangulation by $(\tilde{z}_j - \tilde{z}_i) = w_i w_j (z_j - z_i)$ as in Equation (4.8) if and only if the two planar quadrangulations are discretely conformally equivalent.

This theory of discretely conformally equivalent triangulations and quadrangulations will allow us to check the result of our algorithm presented in Chapter 5 for conformality. The texture's pixel grid naturally creates a polyhedral mesh within the stroke. The pixels are evenly arranged in a rectangular grid of squares, resulting in a cyclic polyhedral quadrangulation. However, it is possible to place diagonals inside the squares, resulting in a triangulation. In Section 5.3.1, we will specify which length-cross-ratios will be used to evaluate the outcome of our algorithm for conformality.

4.2.2 Circle packing

An alternative method for constructing discrete conformal maps based on circle preservation are the so-called circle packings. Thurston introduced the concept of circle packing theory during a conference in 1985, where he proposed that circle packings provide an approximation of continuous conformal maps [Ste99]. This conjecture was subsequently proven in 1987 by Rodin and Sullivan [RS87]. Circle packings discretize the property of conformal maps of preserving infinitesimal shapes (see Definition 1.1.2). They combine the fields of combinatorics and geometry since they are configurations of mutually tangent circles following the combinatorics of a simplicial 2-complex K . This complex K is a triangulation (V, E, F) of an oriented topological surface, which is either the complex plane \mathbb{C} , the unit sphere \mathbb{S} , or the hyperbolic unit disk \mathbb{D} .

A collection of circles P is a circle packing for K if the following holds: there is a circle c_v for each vertex $v \in V$ of K ; for each edge $vw \in E$ of K , there are two circles c_v and c_w that are tangent; and the orientation of triples of circles within P corresponds to the orientation of their corresponding triangular face of K [Ste05, Def. 4.1 & 4.2]. The discrete uniformization theorem [Ste05, Theorem 4.3], [Ste03] guarantees that for every triangulation K , there is a locally finite circle packing P with the same combinatorial structure as K and with nonoverlapping circles in one of the three surfaces \mathbb{C} , \mathbb{S} or \mathbb{D} . It is uniquely determined by K up to conformal equivalence.

Furthermore, the discrete mapping theorem for circle packings states that if K is a simply connected surface, then there exists a locally finite maximal circle packing with mutually disjoint circles in exactly one of the unit disk, the complex plane or the sphere. Additionally, this packing is unique up to conformal automorphisms of the topological surface [Ste03], [Ste05, p.52]. A packing is called maximal if it covers the entire sphere, complex plane or unit circle. In the case of the unit circle, the boundary circles are horocycles, i.e., they are internally tangent to the unit disk (see Figure 4.7 at the top for an example).

Despite the uniqueness theorem, there are multiple circle packings that have the same triangulation K as the underlying complex. In particular for circle packings of the unit disk, mapping one circle packing to another results in various circle configurations [DS95]. Discrete analytic functions between circle packings map circles to circles and preserve

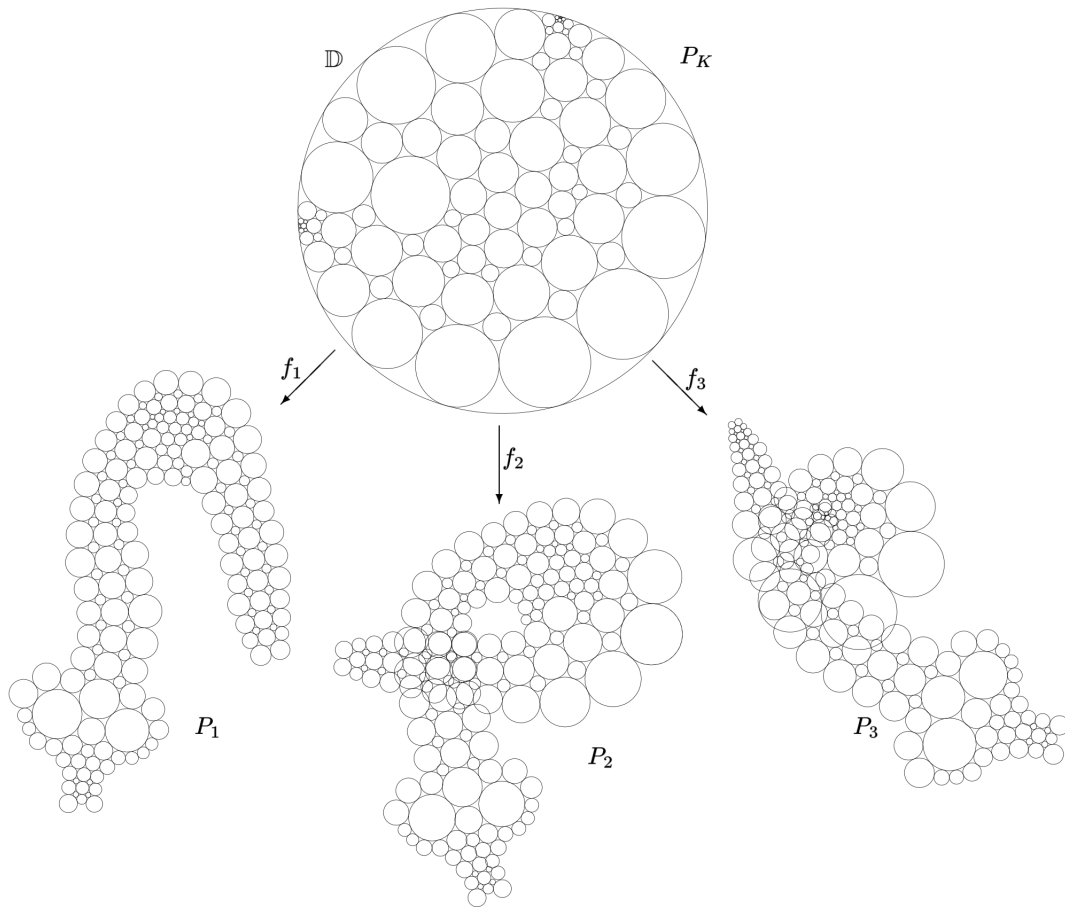


Figure 4.7: Discrete analytic functions mapping a circle packing from the unit disk \mathbb{D} to circle packings with different combinatorics. [DS95, Figure 4]

tangency and orientation. Figure 4.7 shows some examples. The following nomenclature can be found in all cited sources about circle packings.

A packing of tangent circles with mutually disjoint interiors, such as the one depicted on the bottom left in Figure 4.7, is called *univalent*. However, the discrete analytic map may fail to be injective, as illustrated in Figure 4.7 on the bottom in the middle. The circles have disjoint interiors with respect to their neighbors, but the packing as a whole folds over itself, causing certain parts of the region to be covered twice by the circle packing. This type of packing is called *locally univalent*.

This property of local univalence closely resembles self-intersecting strokes. The ornamental strip is meant to be mapped injectively to the stroke, but there are regions of self-intersection, hence of local univalence. The bottom image in the middle of Figure 4.7 serves as a model of self-intersection, which is further discussed in Sections 4.3.1 and 5.4.1.

The third case, displayed in Figure 4.7 at the bottom right, shows a circle packing that is not even locally univalent. There is a branching circle whose tangent neighbors, called petals, wrap around it not just once, but twice or more. The angle sum around each center z_v of the circles in the packing is used to measure this property. For a circle c_v , consider z_v to be its center and p_1, \dots, p_n to be its n petals. The angles at the circle center z_v are denoted as $\alpha_v^{d,d+1}$ and measure the angle between the lines that connect z_v

and two subsequent centers p_d and p_{d+1} , where $n + 1 \cong 1$. The angle sum around z_v is equal to 2π for a (locally) univalent packing, and it is an integer multiple of 2π , i.e., $2\pi \cdot N$ for a branching circle around which the petals wrap $N > 1$ times.

Locating the branch circle in Figure 4.7 is challenging. Instead, it is more convenient to consider the branching circle packing as a circle packing consisting of two sheets, as shown in Figure 4.8. A branch cut of circles is introduced that connects both sheets. Together, the projected sheets define the complete circle packing with branching circle. We will adapt this idea of splitting the packing into two parts in our approach to self-intersecting strokes in Sections 4.3.1 and 5.4.1. Moreover, in Sections 4.3.2 and 5.4.2, we propose ideas on how to process strokes with singular points on their boundary, which includes the concept of splitting the stroke into multiple sheets.

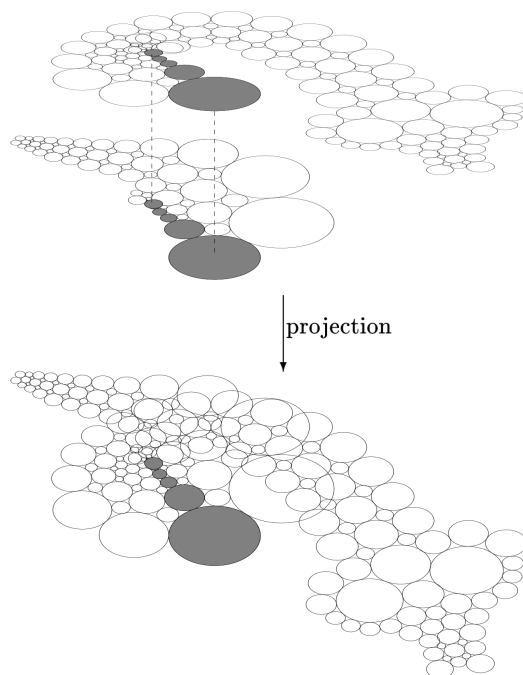


Figure 4.8: A branching circle packing with branch cut. [DS95, Figure 8]

See the work of Stephenson et al. [Ste99], [Ste05], [Ste03], [DS95] and the sources cited therein for algorithms that compute circle packings from combinatorial information and for results on the convergence of discrete circle packings to continuous conformal maps.

4.3 Non-univalent discrete strokes

In the previous section on circle packing, we have demonstrated that self-intersecting strokes possess comparable structures to locally (but not globally) univalent circle packings. For the continuous stroke model, we have examined in detail which singularities occur at the boundary of a stroke and found that the structure of a stroke close to singularities of its boundary is that of a projected swallowtail surface.

For both self-intersections and singular boundary points, the stroke folds over itself, and univalence cannot be achieved, meaning that global injectivity of the expected map from the ornamental strip to the stroke cannot be met. So we will provide algorithms for detecting these two cases of self-intersection and singular boundary points in our discrete model. Detecting these particular cases is the basis for the inclusion of certain procedures (or for concepts thereof) in our algorithm presented in Chapter 5.

4.3.1 Detection of self-intersection

A discrete stroke is defined by an interpolated curve γ , a radius function r , and discrete points $C_{t_d} = \gamma(t_d)$ on γ (see Section 4.1). The stroke is defined as the union of all circles around $\gamma(t_d)$ with radius $r(t_d)$.

To determine whether the newly added data point $\gamma(t_d)$ and its surrounding circle $\mathcal{C}(t_d)$ cause the stroke to intersect itself, we apply the following procedure to the discrete set of all circles $\{\mathcal{C}(0), \dots, \mathcal{C}(t_d)\}$. See Algorithm 2 for the pseudocode of the described algorithm.

For each circle $\mathcal{C}(t_i)$ with $i \in \{0, \dots, d-1\}$, we calculate the distance between its center $\gamma(t_i)$ and the center $\gamma(t_d)$ of the recently added circle. This calculation is necessary to identify circles that intersect the new circle $\mathcal{C}(t_d)$. The distance between the centers of the circles, $|\gamma(t_i) - \gamma(t_d)|$, and the index i are stored in an array.

An entry of the array is deleted if it contains a distance that is greater than the sum of the two radii of the circles $\mathcal{C}(t_i)$ and $\mathcal{C}(t_d)$. The remaining list only contains those distances and indices i in which the centers of the circles are so close to each other that the circles intersect:

$$\|\gamma(t_i) - \gamma(t_d)\| \leq r(t_i) + r(t_d).$$

Subsequently, it is necessary to differentiate between circles within the stroke that are intended to intersect and those that indicate self-intersection of the stroke. All circles $\mathcal{C}(t_i)$ are intentionally constructed to intersect their neighbors because otherwise the “supersonic” Condition (2.2) is not met. To separate these from intersecting non-neighbors, we sort the array by the indices i . If the indices are close to d , the corresponding circles are expected to intersect. Those circles intersecting $\mathcal{C}(t_d)$ with an index i that is not part of a chain of consecutive numbers descending from d indicate self-intersection of the stroke. For instance, if the array still contained entries with indices $d, d-1, d-2, d-25, d-26, d-27$, as shown in Figure 4.9 for a stroke created using the slightly modified Benchmark Example 2.1.8, then the entries with indices $d-2$ and $d-1$ that are consecutive to d belong to circles that are meant to intersect $\mathcal{C}(t_d)$, due to being close neighbors within the stroke. The circles with indices $d-25, \dots, d-27$ indicate a self-intersection of the stroke. Therefore, if any entries in the array contain such non-consecutive indices of d , the stroke will intersect itself when the new circle $\mathcal{C}(t_d)$ is added to the stroke.

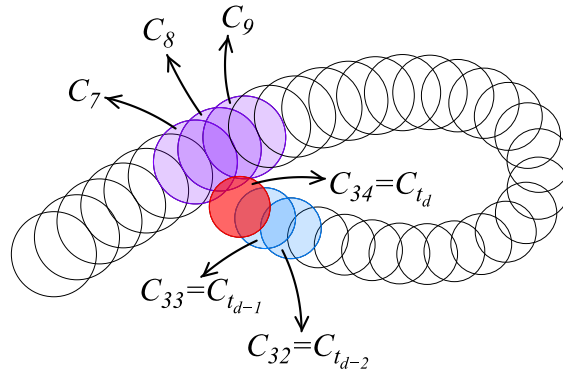


Figure 4.9: Detection of self-intersection of a stroke.

Algorithm 2 Detection of self-intersection of a stroke.

```

array  $\leftarrow$  []
for  $n = 0 \dots d$  do
  index =  $d - n$ 
  distance =  $|\gamma(t_d) - \gamma(t_{\text{index}})|$ 
  if  $r(t_d) + r(t_{\text{index}}) \geq \text{distance}$  then
    append [index,distance] to array
  end if
end for
continue  $\leftarrow$  true
for  $n = 0 \dots \text{length of array}$  do
  if continue then
    if index in the first entry of array equals  $d - n$  then
      remove first element of array
    else
      continue  $\leftarrow$  false
    end if
  end if
end for
if array contains some elements then
  we have detected self-intersection
end if

```

If the stroke makes a narrow turn as in Benchmark Examples 2.1.9 of an ellipse and 2.1.10 of another curve with singular boundary points and non-constant radius function, it also intersects itself because of the folding at those singularities. However, Algorithm 2 will always omit these cases since the folding occurs for circles with consecutive indices. Our observations and concepts concerning discrete strokes with singular boundary points will be presented in the following section.

The effect of detecting a self-intersection on the calculation of the conformal mapping from the strip to the stroke will be discussed in Section 5.4.1.

4.3.2 Identification of singular envelope points

To identify singular points on the stroke boundary, i.e., on the envelope curves w_{\pm} calculated by the interpolated curve γ and function r , we derived a criterion on the curvature of γ in Proposition 2.5.2:

$$\kappa_{\star}(t) = \pm \frac{\|\gamma'(t)\|^2 - r'(t)^2 - r(t)r''(t) + \frac{r(t)r'(t)}{\|\gamma'(t)\|} (\gamma'_x(t)\gamma''_x(t) + \gamma'_y(t)\gamma''_y(t))}{\|\gamma'(t)\|r(t)\sqrt{\|\gamma'(t)\|^2 - r'(t)^2}}. \quad (4.11)$$

Curve γ and the radius function r are both defined as piecewise cubic B-splines, and the stroke is considered not to be continuous, but to be discrete. As outlined in Section 4.1.3, equidistant points C_{t_d} on γ are selected to be centers for the stroke defining circles $\mathcal{C}(t_0), \dots, \mathcal{C}(t_N)$. In each iteration step of this procedure, Equation (4.11) can be computed by inserting the current splines b_d and r_d for γ and r as well as their first and second derivatives. The derivatives of cubic B-splines with respect to t as given in Equation (4.6) can be calculated by taking the derivative of the left component $(1 \ t \ t^2 \ t^3)$. This is because matrix C and the data points $P_{t_{d-1}}, P_{t_d}, P_{t_{d+1}}$ and $P_{t_{d+2}}$ are independent of t . The curvature of the current spline of γ in the iteration corresponds to the curvature of b_d . The curvature is calculated by definition of the curvature of a general curve:

$$\kappa_o(t) = \frac{\det(\gamma'(t), \gamma''(t))}{\|\gamma'(t)\|^3} = \frac{\det(b'_d(t), b''_d(t))}{\|b'_d(t)\|^3}. \quad (4.12)$$

If the right hand sides of (4.11) and (4.12) coincide, the corresponding boundary point $w_{\pm}(t)$ is a singular point. Due to numerical reasons, we cannot enforce equality, but instead require $|\kappa_{\star} - \kappa_o| < \epsilon$ for an appropriately chosen $0 < \epsilon \ll 1$. When choosing ϵ , we have to take into account the step size δ of Algorithm 1. On one hand, ϵ should be chosen sufficiently large to correctly detect each singular boundary point, while ensuring that ϵ is chosen sufficiently small to avoid the identification of multiple “singularities” on w_{\pm} that actually correspond to a single singular boundary point.

Alternatively, we can directly use the facts from which the curvature condition was derived, namely that the boundary and discriminant envelopes of our strokes are identical. According to this, a point on w_{\pm} is a singular point if and only if

$$F(t, x, y) = \frac{\partial F}{\partial t}(t, x, y) = \frac{\partial^2 F}{\partial t^2}(t, x, y) = 0 \quad \text{for } (x, y) = w_{\pm}(t).$$

As shown in Proposition 2.4.1, the points on the envelope curves w_{\pm} automatically satisfy that both, F and its first derivative with respect to t , are equal to zero. Hence, we need to test whether the second derivative at $w_{\pm}(t)$ is zero.

Figure 4.10 shows a stroke and its envelope curves w_{\pm} . For both envelope curves, the values of $\frac{\partial^2 F}{\partial t^2}(t, x, y)$ at $(x, y) = w_{\pm}(t)$ are plotted with respect to t on the right. The two horizontal lines represent the t -axes where $\frac{\partial^2 F}{\partial t^2}(t, w_{\pm}(t)) = 0$. We observe that the second derivative of F w.r.t. t of one envelope curve vanishes at the cusps of the boundary of the stroke.

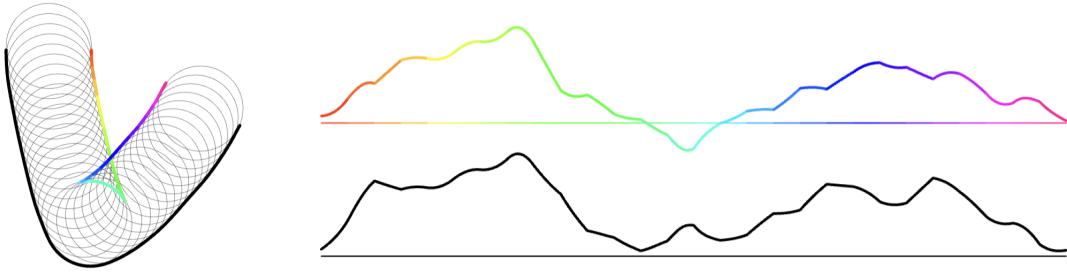


Figure 4.10: A stroke, its envelope curves and the second derivative of F .

Both techniques for identifying singular boundary points fail to differentiate between the two types of singularities: those where the point of regression is a cusp of type A_2 (see Proposition 2.6.8), and those where it is a point of infinite curvature with $A_{\geq 3}$ singularity. Until now, the positions of $A_{\geq 3}$ singularities for strokes with a non-constant radius function r have not been characterized. But from Lemma 2.6.12 we know that in the case of a constant radius function r_c all points of regression on the envelope curves correspond to projections of sections through the cuspidal edges of the swallowtail surface \mathcal{S}_{r_c} . An $A_{\geq 3}$ singularity arises only when the section intersects the surface at the point where its cuspidal edges merge [Tho75, p.65]. It is unlikely to encounter an $A_{\geq 3}$ singularity when drawing a stroke of constant radius by hand. We conjecture that a special treatment for singularities of type $A_{\geq 3}$ can be neglected in the construction of the algorithm.

To provide evidence to support the conjecture, we compute the parameter t^* that corresponds to the curve point of extreme curvature for the ellipse of Benchmark Example 2.1.9. The curvature $\kappa(t^*)$ determines the constant radius $r_c^* = \frac{1}{\kappa(t^*)}$ which is approximately 27 for the curve from Benchmark Example 2.1.9. By construction, the stroke defined by γ , constant radius r_c^* and with envelope curves $\gamma_{\parallel \pm r_c^*}$ has infinite curvature at t^* . We apply our algorithm, which will be introduced in Chapter 5, to this stroke and compute the conformal map without any special handling at the point of infinite curvature (see Sections 5.1 and 5.2 for details). Additionally, we evaluate how well conformality is met; it will be discussed in Section 5.3.1 how this is done. The result is shown in Figure 4.11 for a checkerboard preimage tile. The image is zoomed in to the relevant region of the stroke around the singular boundary point of type $A_{\geq 3}$. The green color throughout the ellipse indicates a good result close to conformality (see details in Section 5.3). In particular, also at the singular boundary point with infinite curvature, where the envelope corresponds to the ellipse's evolute ν shown in orange, the color is green indicating conformality.

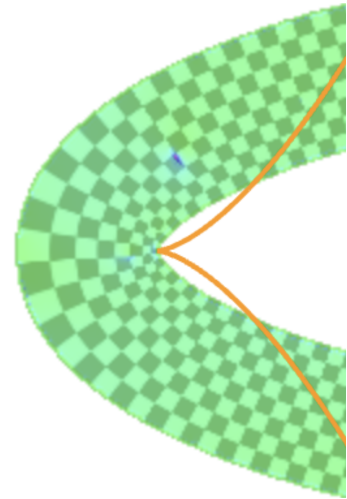


Figure 4.11: Stroke with boundary point of type $A_{\geq 3}$.

For a stroke's curve γ having an extremum of curvature, we have investigated the structure of the corresponding swallowtail surface in Section 2.6.2 for both benchmark examples with constant and non-constant radius function. For strokes with constant radius, cusps only arise on the boundary of a stroke if the curve γ has extreme curvature (see Lemma 2.6.9). For strokes with non-constant radius function, there are more possibilities than this. In both cases, to ensure a mathematically correct representation of the stroke, the folding of the mapped strip should correspond to the structure of a projected swallowtail surface. This is why we investigate this structure in more detail now.

If the stroke neither begins nor ends between two cusps, we know from Section 2.6.2 that they occur in pairs since the two cusps are projections from the two cuspidal edges of a swallowtail surface \mathcal{S} or the respective folds of the surface \mathcal{S}_r for a non-constant radius function.

A pair of cusps S_1 and S_2 is accompanied by two specific points on the projected swallowtail surface: the point S_i where the singular envelope curve intersects itself (the i in S_i stands for intersection), and the point S_e where the projections of the cuspidal edges or the folds of the swallowtail surface meet (the e stands for evolute, we will see the connection in the following). These two additional points, together with the two cusp points, define regions within the stroke that are covered once, twice, or even three times by the projected swallowtail surface. Figure 4.12 depicts a projection of the swallowtail surface that corresponds to the stroke from Benchmark Example 2.1.9 with a constant radius of $r_c = 20$. Figure 4.13 shows the projection of the surface \mathcal{S}_r for curve γ and non-constant radius function r from Benchmark Example 2.1.10.

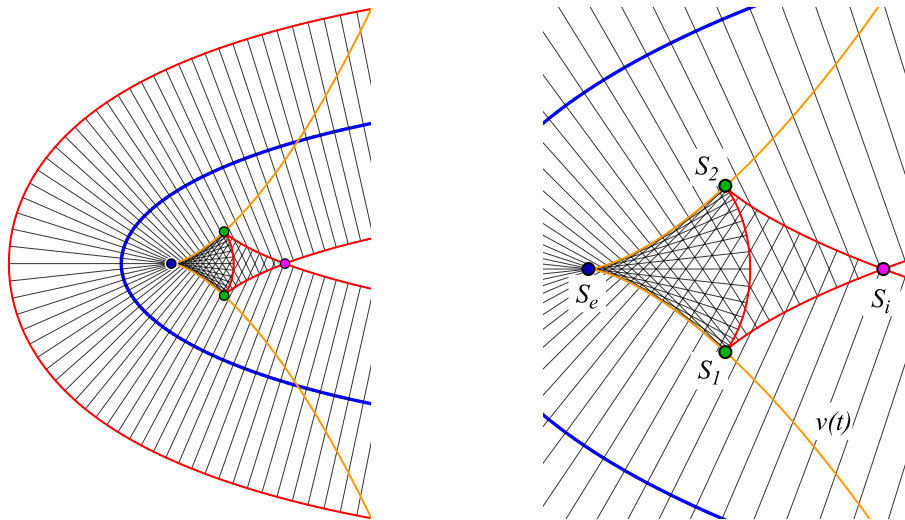


Figure 4.12: Benchmark ellipse 2.1.9 with cusps and folded regions.

We have previously discussed how to identify the locations of the cusp points, S_1 and S_2 , regardless of whether the radius function is constant or not. Likewise in both cases, the doubly covered region of the stroke is the triangular region with vertices being the two cusps S_1 and S_2 as well as the point of intersection S_i of the envelope curve with itself. The edges of the twofold covered triangular region are defined by the envelope curve segments between S_i and S_1 , S_1 and S_2 , and S_2 and S_i , respectively.

To algorithmically determine the location of S_i , we use that S_i is located where the en-

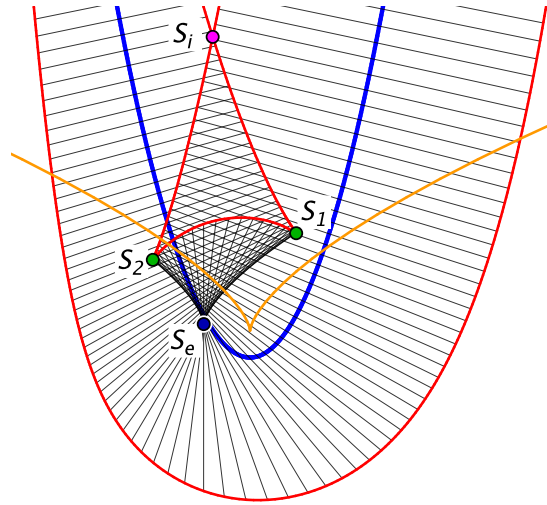


Figure 4.13: Benchmark Example 2.1.10 with cusps and folded regions.

velope curve passes from an already covered region inside the stroke to its outside, crossing itself at S_i . It will be possible to determine the position of S_i locally without computing the intersections of a new boundary spline segment b_d with all previously calculated spline pieces b_j for $j < d$. The exact procedure will be explained in Section 5.4.2 once the structure behind the algorithmic calculations presented in Chapter 5 has been established.

The second special point S_e is the third vertex besides S_1 and S_2 of the threefold covered region within the stroke. It is situated at the intersection of the projected cuspidal edges or folds of the swallowtail surfaces, and hence is located inside the stroke. S_e is connected to the point of curve γ with extreme curvature which occurs between S_1 and S_2 . For this, it can be used as an indicator whether two successive singular points on an envelope curve form a pair or not: if point S_e lies between the two singular points S_1 and S_2 , they form a pair.

If the function has a constant radius, then by Lemma 2.6.9 and Lemma 2.6.12, S_e is the unique point on the evolute ν of γ that corresponds to parameter t^* where γ has extreme curvature between S_1 and S_2 : $S_e = \nu(t^*)$ where $\kappa'(t^*) = 0$. This is precisely the point at which the swallowtail surface shifts from being univalent to threefold (see Section 2.6.2). To determine t^* and thus S_e , the derivative of the radius of curvature is used:

$$\frac{\partial}{\partial t} \frac{1}{\kappa(t)} = \frac{-\kappa'(t)}{\kappa^2(t)}.$$

It vanishes for parameters t where the curvature κ of curve γ has an extremum. As before, we include a test to detect the value of t^* in every iteration of Algorithm 1 by verifying that $\left| \frac{-\kappa'(t)}{\kappa^2(t)} \right| \leq \epsilon$ for an appropriately chosen value $0 < \epsilon \ll 1$, followed by the computation of $S_e = \nu(t^*) = b_d(t^*) + \frac{1}{\kappa(t^*) \|b'_d(t^*)\|} (b'_d(t^*))^\perp$. The curvature and derivative are calculated for the interpolated curve according to Equation (4.12). After identifying vertex S_e , the threefold covered triangular region is bounded by the envelope segment between S_1 and S_2 and two curve segments u_1 between S_e and S_1 and u_2 between S_e and S_2 , respectively. By Theorem 2.3.3, we know that for a constant radius function, the cusp points S_1 and S_2 also

lie on γ 's evolute ν , since all cusps on the envelope curves of strokes with constant radii do. As a result, the projection of the cuspidal edges of the swallowtail surface corresponds to the evolute of γ (see Section 2.6.2). Hence, the curves u_1 and u_2 are segments of ν that intersect at a vanishing angle, creating a cusp at S_e . Thus, the threefold covered region is entirely determined by S_1 , S_2 and S_e , in combination with the envelope curve and the evolute ν .

Lemma 4.3.1 *Given a stroke s defined by a curve γ and a constant radius r_c . If the boundary of s has a pair of singular points S_1 and S_2 , then the vertex S_e of the threefold covered triangular region in s is the cusp of the evolute ν of γ . The two curve segments u_1 between S_e and S_1 and u_2 between S_e and S_2 , which bound the threefold covered region, are segments of the evolute ν .*

If the radius function is not constant, S_e does not generally lie on the evolute of γ , as illustrated in Figure 4.13 for the stroke corresponding to Benchmark Example 2.1.10, with the evolute depicted in orange. Nevertheless, when it comes to the surface \mathcal{S}_r of the stroke in Benchmark Example 2.1.10, we observe that S_e is related to the curve point of extreme curvature. The point S_e is determined by finding the intersection point of two line segments. The first line segment connects $w_+(t^*)$ and $w_-(t^*)$ while the second connects $w_+(t^* + \epsilon')$ and $w_-(t^* + \epsilon')$, where ϵ' is a positive value less than the step width between two equidistant curve points. So far, it remains unclear whether this calculation is generally applicable. We leave this as an interesting open question.

With this chapter, we completed the preliminary theoretical work for our algorithm. Also, we are able to compute all relevant components of the strokes during the drawing process, like the interpolated curve γ , the interpolated radius function r , the envelope curves w_{\pm} and the reference points for the exceptional cases of self-intersections and singular boundary points. Hence, we are now ready to develop the algorithm in the next chapter.

5 | Implementation: Pixel Averaging on the GPU

In this chapter, we will present and discuss our algorithm that computes the conformal image of the artistic content from a strip consisting of multiple copies of an ornamental tile onto a user-drawn stroke. The stroke is modeled by a family of circles defined by discrete points on the user-drawn path and their associated radii. The ornamental strip is mapped to the stroke by our algorithm in real-time during the drawing process.

Parts of this chapter may be found in our previously published paper [Pol23a].

The general workflow of the algorithm starts with the initialization of preimage coordinates for all points in the stroke, which are provided by the preimages of the circles that define the stroke. Then, by iteratively averaging the preimage coordinates simultaneously to the drawing process, the ornamental image in the stroke converges to a nearly conformal result. The computational process is made visible in such a way that the artistic content of the strip flows to its final position within the stroke as it is drawn.

This real-time requirement of our algorithm is an important difference to other algorithms that calculate conformal maps and their approximations, as they usually display the solution once all the calculation steps are complete. To fulfill this real-time constraint, we have decided to implement our algorithm on the graphics processing unit (GPU), one of the computational units of modern computers, which is able to perform fast computations on graphical output. The following statements concerning the nature of the GPU follow arguments of Montag [Mon20] and Owens et al. [OLG⁺07].

The commonly used processor for computer programs is the central processing unit (CPU), which is designed for object-based sequential computations, while the GPU works massively in parallel. The difference can be described as follows: to draw all stroke defining circles $\mathcal{C}(t_d)$ with centers $\gamma(t_d)$ and radius $r(t_d)$ for $d \in \{0, \dots, N\}$ to the drawing surface, the CPU takes a command like

draw the circle with radius $r(t_d)$ and center $\gamma(t_d)$ for d from 0 to N .

The GPU, however, is not object based but pixel based, i.e., it works locally on every pixel of the texture underlying the drawing surface, which makes the code rather something like

test for each pixel of the drawing surface whether it has a distance of $r(t_d)$ to $\gamma(t_d)$ for all $d \in \{0, \dots, N\}$, and if this is the case, color this pixel in the desired color of the circle.

This closely connects the computations to the graphical textures, which serves our goal of showing the computational steps of the algorithm. It is also consistent with our goal

of computing a nearly conformal map, which is locally a similarity transformation, that the computations are performed locally. Writing code for the GPU involves converting all calculations into graphical terms, making it more than just learning a new programming language. This code may appear less intuitive because the GPU processes the pixels of a texture in parallel, without a predetermined order.

The speed of GPU computations depends on the available parallel units of the GPU, i.e., how many pixels can be processed in parallel. Additionally, the speed of GPU computations is influenced by the number of data transfers between the CPU and GPU, as well as the amount of pixels in the considered texture. To minimize the number of required data transfers between the GPU and the CPU, we will focus on calculations based on textures and the use of the GPU in the following. In this sense, all computational steps will be stored in textures from which they can be retrieved and used for additional computations.

A two-dimensional texture consists of a finite number of pixels arranged in a rectangular grid. We will mainly use textures of 1024×512 pixels, since our experiments have shown that this is a reasonable number of pixels that does not make the calculations too slow and also supplies us with a resulting image of decent resolution. We denote the coordinates of a pixel by (x, y) , where $x \in [0, 1024]$ and $y \in [0, 512]$, i.e., our main textures covering the drawing surface have vertices $(0, 0)$, $(1024, 0)$, $(0, 512)$ and $(1024, 512)$. These pixel coordinates may also take values that do not precisely coincide with the positions of the pixels in the texture, which will be discussed in Section 5.1.3.

Every pixel p in a texture is assigned a 4-dimensional vector $(r, g, b, a)_p$. The four entries are called the red-, green-, blue- and alpha-channels, which intuitively makes sense for the graphical output of a texture: (r, g, b) gives the color information for each pixel, and the alpha channel contains the pixel's opacity, i.e., a value that indicates how transparent the color in the first three entries is displayed at the corresponding pixel. In our setup using *CindyJS*, each of the entries is an 8-bit integer between 0 and 255, which is used for color representation by default. If the pixels of a texture are intended to store intermediate steps for more complex computations, as it will be the case in our algorithm, the 8 bits may not be enough. How to address this issue will be discussed in Sections 5.1.1 and 5.1.2.

In the following, we will present the algorithm that provides us with a nearly conformal map from an ornamental tiled strip to a user-drawn stroke. For this, we will temporarily restrict our strokes to be free of self-intersections and singular boundary points. These special cases will be discussed at the end of this chapter in Section 5.4. For our implementation, we use the language *CindyJS* [RGK23b] together with the plugin *CindyGL* [M⁺17]. *CindyJS* is a framework that enables the creation of interactive applets embedded in HTML files for use in web browsers. The *CindyGL* plugin provides a higher level interface to WebGL, an open web standard for rendering graphics in the browser without additional plugins. *CindyGL* allows the user to write *CindyJS* code that is passed to WebGL. All code snippets in the following are either given in pseudocode or in *CindyJS* code. The entire code for the HTML applet, titled *OrnaStrokes_LP.html* and written in *CindyJS*, can be accessed through [Pol23b].

In Section 5.1, we will introduce the general implementation framework. We will present how the initial preimage coordinates for all pixels in the stroke are computed from

the registered input data (Section 5.1.1). Also, the color transfer from the ornamental strip to the stroke via reverse pixel lookup will be discussed (Section 5.1.3). To enable GPU computation, the required data for the algorithm is stored on textures, as will be outlined in Section 5.1.2.

In Section 5.2, we will present the core algorithm of the conformal computations. For each pixel p within the stroke, the algorithm calculates the average of the neighboring pixel’s preimage coordinates to obtain preimage coordinates for the current pixel p (Section 5.2.1). If any neighboring pixels necessary for the pixel averaging process are located outside the stroke, p is said to be in the boundary area. These pixels are subsequently processed using our results from Section 3.3, which covers the extension of conformal maps to the boundary of the respective domains and beyond. The Schwarz reflection principle, along with its generalization, is applied to our parallelized GPU computations to treat pixels in the boundary area (Section 5.2.2).

In Section 5.3.1, we will demonstrate that the algorithm accurately computes an approximately conformal map. Based on the definitions of conformality presented in this thesis, a local measure of the map’s deviation from conformality will be given. The representation of deviations as color values in a texture supports the observation that our algorithm closely approximates conformality (Section 5.3.1). A global measure will be defined that is used as a stopping criterion for the averaging procedure. This global measure computes the average of all deviations per pixel in the stroke (Section 5.3.2). To compare our method with an exact conformal map, we will calculate the conformal modulus of a half annulus and the rectangle to which it is conformally mapped. We will compare it to the conformal modulus of the rectangular strip mapped to the stroke of Benchmark Example 2.1.7 (Section 5.3.3).

Section 5.4.1 examines strokes with self-intersection. We will present the necessary adaptations to the general algorithm, the pixel averaging, and the conformal test. Finally, we will propose methods for integrating the case of singular boundary points into our algorithm (Section 5.4.2).

5.1 General algorithm

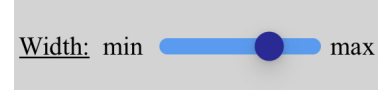
In this section, we outline the general algorithm for mapping the tiled strip onto the stroke, along with some interesting implementation details. The general algorithm will be completed in the following sections by a pixel averaging algorithm, a method to verify conformality, and techniques for self-intersecting strokes and singular boundary points.

The algorithm is initiated when the user places a mouse, finger or digital pen on the drawing surface. From then on, the current stroke is drawn on the drawing surface. The underlying texture that contains the corresponding color data for every pixel in the stroke is called `currentCanvas`. Once the user stops drawing, i.e., the mouse, finger, or digital pen is lifted from the drawing surface and no further data is registered, the appearance of the stroke is fixed. This means that only the calculations concerning the displayed colors continue. If a new stroke is initiated, the texture `currentCanvas` is cleared and the previously drawn stroke is transferred to a texture called `storeCanvas`.

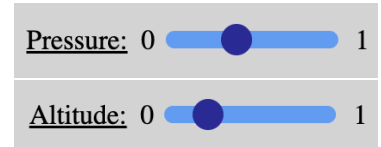
To model the stroke, we have already seen in Section 4.1.3, Algorithm 1, how the centers of the circles $\mathcal{C}(t_d)$ are determined as equidistant points $\gamma(t_d)$ on the interpolated

user-drawn path. The radius $r(t_d)$ of circle $\mathcal{C}(t_d)$ is determined depending on the registered pressure of the pen on the drawing surface and probably other available data.

If the user draws with a mouse on a computer screen or with a finger on a tablet, no pressure sensitivity influences the stroke width. Instead, the user interface contains a slider that allows the user to specify the constant radius for the circles of the upcoming stroke. For our purposes, we limit the radius value between a minimum of 10 and a maximum of 40. Based on our investigations, decreasing the radius below 10 significantly reduces the recognition of the artistic content from the input tile T .



If the user draws with a pressure-sensitive digital pen, various data is registered as, for instance, the amount of pressure applied or the angle at which the pen is tilted towards the drawing surface, also known as the altitude angle. Two additional sliders control the influence these two properties have on the radius value. We calculate the radius value by combining the information from all three sliders using the following formula:



$$r(t_d) = 40 \cdot \left((1 - p\%) * \left(a\% * (1 - \alpha_r \cdot 2/\pi) + (1 - a\%) \cdot r_c/40 \right) + p\% \cdot p_r \right) \quad (5.1)$$

for $r_c \in [10, 40]$, $p\% \in [0, 1]$ the influence of the pressure information deduced from the respective slider, $p_r \in [0, 1]$ the measured pressure, $a\% \in [0, 1]$ the influence of the pen's altitude angle and $\alpha_r \in [0, \frac{\pi}{2}]$ the measured tilt of the pen towards the drawing surface. The slider value $p\%$ determines the percentage with which the value $r(t_d)$ depends on the applied pressure. In the case that $a\% \neq 0$, which means that the stroke width also depends on the tilt of the digital pen, the constant component $(1 - p\%) \cdot r_c/40$ is reduced. If the value calculated by Equation (5.1) is less than 10, the resulting radius value $r(t_d)$ is again set to 10.

When the user starts drawing, the first registered position $\gamma(t_0)$ on the path together with the determined radius value $r(t_0)$ defines the first circle $\mathcal{C}(t_0)$ of the stroke. All subsequent circles are determined once the piecewise interpolated curve γ and radius function r enable the calculation of equidistant circle centers (see Section 4.1). In Section 5.1.1, we will specify the initial coordinates for the pixel averaging algorithm presented in Section 5.2 from these equidistant curve points $\gamma(t_d)$ and radii $r(t_d)$. When the user stops drawing, no more data is registered and the radius function r is set to zero to indicate that the appearance of the stroke is fixed.

5.1.1 Initial coordinates for pixel averaging

Given a rectangular tile T with width w_T and height h_T , the goal is to compute a conformal map $f: S_n \rightarrow s$ from the finite strip S_n defined by n copies of T to the stroke s . To obtain this map f , we actually compute the inverse map $f^{-1}: s \rightarrow S_n$, i.e., we determine the preimage coordinates in the tiled strip for each pixel within the stroke. In this section, we present how to compute the initial coordinates, which are then used to iteratively obtain the conformal map f from the strip to the stroke. We store the coordinates in

textures (Section 5.1.2) and calculate the desired conformal map or rather its inverse f^{-1} using a pixel averaging algorithm (Section 5.2).

Initial preimage coordinates for all pixels inside the circles $\mathcal{C}(t_d) \in s$ are calculated once the circles are registered. For this purpose, we define similarity transformations $\tau_d: \mathcal{C}(t_d) \rightarrow S_n$, which consist of rotation, scaling, and translation. Hence, the preimage coordinates in S_n also lie within circles, and the preimage region of the stroke consists of a rectangle with two half circles on either side (see Figure 5.1 containing additional information used in Section 5.3.3).

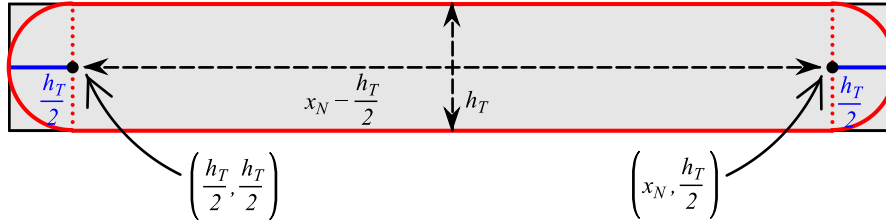


Figure 5.1: The preimage region of a stroke in the rectangular strip S_n .

Structurally, the aim is to map the boundaries of the preimage region in the strip and the stroke to each other. Hence, the initial preimage coordinates are defined in such a way that all circles $\mathcal{C}(t_d)$ are mapped to circles of diameter h_T fitting the strip. This implies that all centers of preimage circles will be situated on the horizontal line $y = \frac{h_T}{2}$ in S_n . All circle centers within the stroke lie on the interpolated curve γ , which parameterizes this curve equidistantly. Thus, the maps τ_d for initial preimage coordinates map the equidistant parametrization of curve γ to the horizontal line $y = \frac{h_T}{2}$. The initial circle $\mathcal{C}(t_0)$ is mapped onto the circle centered at $\tau_0(\gamma(t_0)) = (\frac{h_T}{2}, \frac{h_T}{2})$ with a radius of $\frac{h_T}{2}$. To determine the location of $\tau_d(\gamma(t_d))$ on the line $y = \frac{h_T}{2}$ within the rectangular strip S_n , we scale the distance between two consecutive points on the curve, $\gamma(t_d)$ and $\gamma(t_{d-1})$, while considering the diameters of both the target circle, which has a diameter of h_T , and the circle $\mathcal{C}(t_d)$, which has a diameter of $2 \cdot r(t_d)$:

$$\begin{aligned} \tau_0(\gamma(t_0)) &= \left(\frac{h_T}{2}, \frac{h_T}{2} \right) \\ \tau_d(\gamma(t_d)) &= \left(x_d, \frac{h_T}{2} \right), \quad x_d = x_{d-1} + \frac{h_T \cdot |\gamma(t_d) - \gamma(t_{d-1})|}{2 \cdot r(t_d)} \end{aligned} \quad (5.2)$$

We aim for f^{-1} to map the envelope curves of the stroke to the upper and lower horizontal lines bounding the tiled strip S_n . To achieve this, we define the preimages under τ_d of these lines $y = 0$ and $y = h_T$ to be the parallels to the tangent at $\gamma(t_d)$. These tangents have a direction vector of $\gamma'(t_d)$ and are positioned at a distance of $\pm r(t_d)$ from $\gamma(t_d)$:

$$\gamma(t_d) - r(t_d) \frac{(\gamma'(t_d))^\perp}{\|\gamma'(t_d)\|} + \lambda \cdot \gamma'(t_d) \mapsto y = 0 \quad \gamma(t_d) + r(t_d) \frac{(\gamma'(t_d))^\perp}{\|\gamma'(t_d)\|} + \lambda \cdot \gamma'(t_d) \mapsto y = h_T.$$

The map τ_d is defined by three vertices P_1, P_2, P_3 of a square around $\mathcal{C}(t_d)$, which is parallel to the tangent at the center $\gamma(t_d)$, and by their three preimage vertices in the strip as shown in Figure 5.2. All six vertices are given by

$$\begin{aligned} P_1 &= \gamma(t_d) - \frac{r(t_d)}{\|\gamma'(t_d)\|} (\gamma'(t_d) + \gamma'(t_d)^\perp) & \mapsto & \tau_d(P_1) = \tau_d(\gamma(t_d)) + \frac{h_T}{2}(-1, -1) \\ P_2 &= \gamma(t_d) + \frac{r(t_d)}{\|\gamma'(t_d)\|} (\gamma'(t_d) - \gamma'(t_d)^\perp) & \mapsto & \tau_d(P_2) = \tau_d(\gamma(t_d)) + \frac{h_T}{2} \cdot (1, -1) \\ P_3 &= \gamma(t_d) - \frac{r(t_d)}{\|\gamma'(t_d)\|} (\gamma'(t_d) - \gamma'(t_d)^\perp) & \mapsto & \tau_d(P_3) = \tau_d(\gamma(t_d)) + \frac{h_T}{2} \cdot (-1, 1) \end{aligned} \quad (5.3)$$

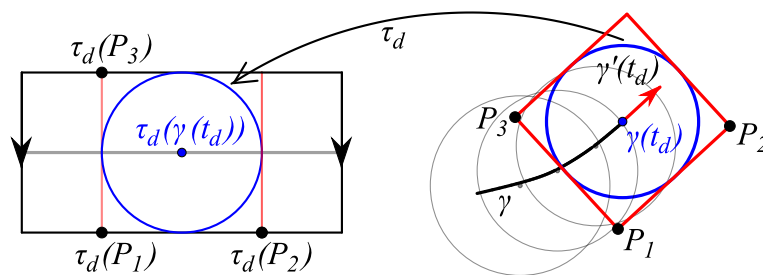


Figure 5.2: The local map τ_d as initial map from the stroke to the strip.

These three pairs of points uniquely define τ_d , and for each pixel p within $\mathcal{C}(t_d)$, its preimage coordinates $\tau_d(p)$ in the ornamental strip can be calculated. Since the supersonic Equation (2.2) defines subsequent circles to overlap, a pixel p within multiple circles $\mathcal{C}(t_d)$ could potentially be mapped with different maps τ_d . To solve this, we only assign new preimage coordinates to a pixel if it has not been assigned a preimage point before, i.e., if p lies within the crescent of the newly added circle that does not overlap with the rest of the stroke. We will discuss this in more detail in Section 5.1.2.

As the number of circles $\mathcal{C}(t_d)$ increases, so does the number of preimage circles within the strip S_n . The coordinates $\tau_d(p)$ progressively increase in their x -coordinates, as indicated by the preimage coordinates of the circle center $\tau_d(\gamma(t_d))$ in Equation (5.2). However, the tiled strip S_n is known to consist of equivalent tiles T concatenated with each other. Hence, it is not necessary for the preimage coordinates to be located across the entire strip. It is sufficient to take $m \leq n$ copies of T and calculate the coordinates of $\tau_d(p)$ modulo $m \cdot h_T$ within the shortened strip S_m .

We need to take enough copies of T so that two complete circles can be placed next to each other on S_m . Since each preimage circle has a diameter of h_T , which is the height of T and therefore of S_n and S_m , let m be defined as

$$m = \begin{cases} 2 \cdot \frac{w_T}{h_T} & \text{if } \frac{w_T}{h_T} \geq 1 \\ 2 \cdot \frac{w_T}{h_T} \cdot \lceil \frac{h_T}{w_T} \rceil & \text{if } \frac{w_T}{h_T} < 1 \end{cases}.$$

In either case, m is greater than or equal to 2, which makes S_m at least twice as wide as it is high, so that two circles of diameter h_T can be placed tangentially side by side

in S_m . We identify the vertical segments on the left and right sides of strip S_m . In other words, S_m topologically is a cylinder representing the modulo computations of the preimage coordinates.

As a consequence, the maps τ_d must also be computed within this smaller preimage strip using modulo calculations. If the x -coordinate of $\tau_d(P_2)$ is smaller than that of $\tau_d(P_1)$ modulo $m \cdot h_T$, the map τ_d maps the square around the circle $\mathcal{C}(t_d)$ to a square inside S_m of opposite orientation along the x -axis. To solve this, before defining and applying τ_d to $\mathcal{C}(t_d)$, the coordinates of the preimage points $\tau_d(P_1)$, $\tau_d(P_1)$, and $\tau_d(P_1)$ are adjusted by a shift of $-m \cdot h_T$.

When the preimage coordinates will be used for further computations in Section 5.2, it must be considered that these coordinates are given modulo $m \cdot h_T$.

5.1.2 Storing coordinate information on textures

As outlined earlier, we compute the preimage coordinates $\tau_d(p)$ for all pixels p contained within the circles $\mathcal{C}(t_d)$ of the stroke. We use the GPU to ensure fast and local computation of the conformal map f between the tiled strip S_n and the stroke s . Since transferring memory between the GPU and the CPU is expensive and slows down calculations, we store all the information needed for future calculations in GPU memory as textures. This means that we define several textures covering the drawing surface and store different pieces of information on these textures for each pixel in the stroke. All of the textures listed below contain 1024×512 pixels and are placed at $(0, 0)$ as the lower left corner, parallel to the coordinate axes. All *rgba*-vectors are initially set to $(0, 0, 0, 0)$, but for each pixel p within the stroke, the textures will contain

- `preimageCanvas`: ... the preimage $\tau_d(p)$ of the pixel within the tiled strip.
- `centerCanvas`: ... the center $\tau_d(\gamma(t_d))$ of the corresponding preimage circle of $\mathcal{C}(t_d)$.
- `centerStrokeCanvas`: ... the center $\gamma(t_d)$ of the circle $\mathcal{C}(t_d)$.
- `radiusCanvas`: ... the radius $r(t_d)$ of the circle $\mathcal{C}(t_d)$.
- `equiCanvas`: ... the sequential number d of the discrete step pixel p belongs to.

As previously discussed, the circles $\mathcal{C}(t_d)$ intersect due to the supersonic Condition (2.2), which makes the preimage coordinates non-univalent. Hence, it is necessary to decide which data to store for each pixel p . For the first circle $\mathcal{C}(t_0)$, all pixels with a distance of at most $r(t_0)$ from $\gamma(t_0)$ on all textures are assigned the corresponding information. For all further circles $\mathcal{C}(t_d)$ with $d > 0$, the pixels p within $\mathcal{C}(t_d)$ on each of the above textures that do not contain any information so far, i.e., the pixels $p \in \mathcal{C}(t_d) \setminus \mathcal{C}(t_{d-1})$, store the information related to the circle $\mathcal{C}(t_d)$. Visually speaking, for every new circle $\mathcal{C}(t_d)$, the previously uncovered circular crescent is added to the stroke (see Figure 5.3).

Note 5.1.1 *It is possible to update stored information for all pixels in a recently added circle. However, this results in constant redefinition of the preimage coordinates on the last part of the stroke in drawing direction, which interrupts the ornamental flow to the conformal solution.*

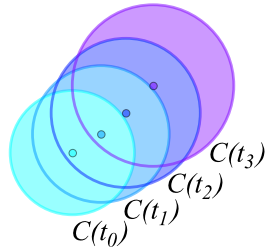


Figure 5.3: Circular crescents are iteratively added to the stroke.

The question arises of how to determine if a pixel already belongs to the stroke and how the respective information is stored in each texture. For this, we recall that each pixel p is assigned a 4-dimensional vector $(r, g, b, a)_p$ which has 8-bit entries, i.e., each of the red-, green-, blue- and alpha-channel can store 256 different values, as determined by our calculations with CindyGL.

However, storing precise coordinate information within each vector's (r, g, b, a) entries requires more than the 256 different numbers per channel. Our method of reducing the size of the stored coordinate information by calculating all preimage coordinates modulo $m \cdot h_T$ is not yet sufficient to overcome this storage problem. This is why we divide all information into separate vector entries. It is also possible to split the information into multiple textures, but this would slow down the calculations.

For this, we use the function `splitData` given in Algorithm 3 [RG21]. The function stores the preimage coordinate vector $(x, y) = \tau_d(p)$ for a pixel p in all four vector entries of the *rgba*-vector of that pixel. Instead of storing 256 different integers in the 8 bits per entry of the *rgba*-vector, we rescale the data to the interval $[0, 1]$, which enables us to store 256 equally spaced fractional numbers. To obtain an accurate fraction that can be stored in an 8-bit entry with minimal information loss, the scaled values are multiplied by the base value of 256, rounded to an integer, and divided by the base of 256. The remainder is treated the same way and stored in the other two entries of the *rgba*-vector at the corresponding pixel p of the `preimageCanvas`.

Algorithm 3 Function storing two coordinates in four *rgba*-entries for higher precision.

`scvel` $\leftarrow m \cdot h_T$

`base` $\leftarrow 256$

splitData $((x,y), \text{scvel}, \text{base}) :=$

$u_x \leftarrow \max\left(\min\left(\frac{x}{\text{scvel}}, 1\right), 0\right)$

$u_y \leftarrow \max\left(\min\left(\frac{y}{\text{scvel}}, 1\right), 0\right)$

$u_{x_1} \leftarrow \text{round}(u_x \cdot \text{base}) \cdot \frac{1}{\text{base}}$

$u_{y_1} \leftarrow \text{round}(u_y \cdot \text{base}) \cdot \frac{1}{\text{base}}$

$u_{x_2} \leftarrow \text{round}\left(\left(u_x - u_{x_1}\right) \cdot \text{base}^2\right) \cdot \frac{1}{\text{base}}$

$u_{y_2} \leftarrow \text{round}\left(\left(u_y - u_{y_1}\right) \cdot \text{base}^2\right) \cdot \frac{1}{\text{base}}$

return $(u_{x_1}, u_{y_1}, u_{x_2}, u_{y_2})$

The code in CindyJS for writing on `preimageCanvas` is given in Algorithm 4. Here, the new center $\gamma(t_d)$ on the curve γ is designated as `newcenter`, while `radval` represents

the value of the radius function at t_d . Additionally, `mat` denotes the transformation matrix of the similarity τ_d that is defined by three preimage and image points as given by Equations (5.3). Finally, `scvel` is equal to $m \cdot h_T$, while `base` is equal to 256, as previously stated. The `colorplot` command runs the code in its fourth entry on the GPU and stores the solution in the pixels of the given texture at the specified position. The command `imagergba` reads the `rgba`-vector of the texture at the pixel in the fourth variable.

Algorithm 4 Writing to a texture using the `colorplot` command.

```

1: colorplot((0, 0), (1024, 0), "preimageCanvas",
2:   p=#.xy;
3:   rgba =imagergba((0, 0), (1024, 0), "preimageCanvas", p );
4:   ref=imagergba((0, 0),(1024, 0), "centerCanvas", p );
5:   pixeldist= (newcenter_1-p_1,newcenter_2-p_2);
6:   preimpt=(mat ·(p_1,p_2, 1)).xy;
7:   if( | pixeldist | ≤ radval+0.5, // if pixel p is within the new circle
8:     if(ref_4 < 1 , //if the pixel p is in the newly added circular crescent
9:       rgba =splitData((preimpt_1,preimpt_2, 0, 1),scvel,base);
10:    ););
11: rgba ); // the rgba-vector to the pixel p

```

The information saved in the other textures is handled by modified versions of the function `splitData`. The preimage circle center $\tau_d(\gamma(t_d))$ is stored in `centerCanvas` using `splitData` reduced to only one coordinate since the y -coordinate within S_m is always $\frac{h_T}{2}$. The circle center $\gamma(t_d)$ is stored in `centerStrokeCanvas` by the original function `splitData`, but for a scaling value of 1024, since the x -coordinate is at most 1024. The value of the radius function $r(t_d) \in [10, 40]$ is stored in `radiusCanvas`. Although the value is small enough to be directly stored, `splitData` is still applied to $(r_1, r_2) = (\lfloor r(t_d) \rfloor, (r(t_d) - r_1) \cdot 100)$ with `scvel=base= 256` to capture precise decimals. The number of equidistant circles d is stored in `equiCanvas` dividing d into $(d_1, d_2) = (d \bmod 256, \lfloor \frac{d}{256} \rfloor)$ and applying `splitData` for `scvel=base= 256`.

To use the stored coordinates for further computations, we need the inverse function of `splitData` and of its modifications:

$$(\tilde{x}, \tilde{y}) = \text{restoreData}((u_{x_1}, u_{y_1}, u_{x_2}, u_{y_2})) = \text{scvel} \cdot \left(u_{x_1} + \frac{u_{x_2}}{\text{base}}, u_{y_1} + \frac{u_{y_2}}{\text{base}} \right). \quad (5.4)$$

Due to the rounding and the limited number of fractions that can be stored by the 8-bit channels, some information may be lost in (\tilde{x}, \tilde{y}) compared to the original (x, y) . However, this loss of information is negligible and does not affect the process.

The texture `centerCanvas` is not only essential for storing the center of the preimage circle. Also, we define that it contains the value 1 in its alpha-channel for all pixels inside the stroke and a zero for all pixels outside the stroke. This is possible because only two entries per `rgba`-vector are used to store the x -coordinate of $\tau_d(\gamma(t_d))$. Therefore, the `rgba`-vector of a pixel p in `centerCanvas` serves as an indicator for determining whether the said pixel is inside or outside of the stroke. For this reason, the texture `centerCanvas`

is always updated after all other textures, so that the fourth coordinate of a pixel p in the new circular crescent $\mathcal{C}(t_d) \setminus \mathcal{C}(t_{d-1})$ indicates that the other textures do not yet contain the relevant information at pixel p .

5.1.3 Reverse pixel lookup

With the textures from the previous section, we can determine which pixels are inside the stroke and which pixels are outside by checking the alpha-channel value in `centerCanvas`. So we can draw the simulated stroke on `currentCanvas` by assigning a color vector $(r, g, b, 1)$ to all pixels p in stroke s . However, we are not interested in a monochrome stroke, but in a stroke that displays the content of the repeatedly conformally mapped ornamental tile T . Until now we have calculated the initial preimage coordinates for all pixels within the stroke, i.e., we know for p in the stroke s that its preimage in the reduced tiled strip S_m of the ornamental tiles T has the coordinates encoded at position p in the texture `preimageCanvas`. Thus, we can use the function `restoreData` from Equation (5.4) to access the preimage coordinates for all pixels in the stroke and look up the color that this coordinate has in the ornamental tile T . This color is then passed to the texture `currentCanvas` which displays the stroke on the drawing surface. This method is called *reverse pixel lookup*. The corresponding CindyJS code is given in Algorithm 5 making use of the texture `tileCanvas`, in which the m copies of the tile T are stored.

Algorithm 5 Reverse pixel lookup from the tile texture.

```

1: colorplot((0, 0), (1024, 0), "currentCanvas",
2:   p=#.xy;
3:   preim=imagergba((0, 0), (1024, 0), "preimageCanvas", p, interpolate→ false);
4:   coord=restoreData(preim,scvel,base);
5:   ref=imagergba((0, 0), (1024, 0), "centerCanvas", p, interpolate→false);
6:   if(ref_4 > 0, //if the pixel p is in the stroke
7:     col=imagergb((0, 0), (1024, 0), "tileCanvas", (coord_1, coord_2));
8:   , // else if pixel p is not in the stroke
9:     col=(0, 0, 0, 0); // the pixel is drawn transparent
10:  );
11: col ); // store the rgba-vector col to the pixel p

```

Algorithm 5 uses the modifier `interpolate→ false` in the function `imagergba`. If `interpolate` is `true`, the `imagergba` function called at p returns the linearly interpolated value of the four neighboring pixels of p ; if `interpolate` is `false`, `imagergba` returns the value of the pixel closest to p [M⁺17]. In our setup, where the pixel vectors contain precise preimage coordinate information, we need to read the exact *rgba*-value that was stored at the requested position p . Therefore, we have to set `interpolate` to `false`.

Figure 5.4 displays the result of the reverse pixel lookup at this point in the algorithm. Clearly, this is not the result we are looking for. But we can imagine what a conformal map of the tiled strip to the stroke might look like if the preimage coordinates are adapted accordingly. This illustrates that the prior selection of preimage coordinates provides a good starting point for our pixel averaging algorithm, which operates on the preimage coordinates and is presented in the following section.

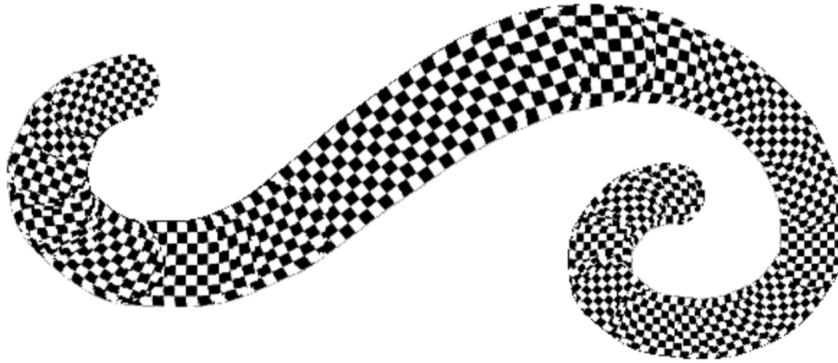


Figure 5.4: The visual result of the initialization of preimage coordinates in a stroke.

Actually, when transferring the color from the ornamental tile T to the stroke based on the preimage coordinates stored in `preimageCanvas`, we are performing a transfer of the underlying coordinate system from the tile to the stroke. This will be relevant again later.

5.2 Pixel averaging

In this section, we present the essential part of our real-time computations of the conformal map f from the tiled strip S_m to the stroke s . We have initial preimage coordinates $q = (\tilde{x}, \tilde{y})$ for all pixels $p = f(q)$ in stroke s stored in the texture `preimageCanvas`. Now, we apply the principle of *pixel averaging* to them. Often, pixel averaging is used for blurring effects of images or for similar results. We perform the averaging at the level of the preimage coordinates q , which will actually converge to an approximately conformal map f (see Section 5.3). The averaging is applied simultaneously to all pixels of `preimageCanvas` and it is repeated multiple times per drawing event. Since the reverse pixel lookup displays the colors read from tile T to the stroke based on the current coordinates stored in `preimageCanvas`, the coordinates visually flow into their intended position in real time as the stroke is drawn.

A similar algorithm implemented on the CPU is presented by Swart [Swa11]. It takes planar preimage and target domains as input and computes the average of the preimage coordinates of four neighboring pixels for each pixel in the target domain. In each step, a randomly selected pixel in the target mesh is treated. If a pixel has fewer than four neighbors within the domain, it is considered to be on the boundary and an adapted result of the averaging step is shifted to the closest position on the boundary. Koczyński et al. also use an averaging algorithm based on four neighbors to map a band model of the hyperbolic plane to arbitrary simply connected domains with long, narrow regions [KCK19]. They fix two positions on the boundary of the target domain, which then act as the images of $\pm\infty$, i.e., the limits of the band model. This divides the target boundary into two parts, to which the upper and lower bounding lines of the hyperbolic band model are to be mapped. If pixels within the target domain have neighboring pixels on the boundary, they are skipped in the averaging process proposed by Koczyński et al.

The fundamental GPU-based pixel averaging algorithm was developed by Montag and Richter-Gebert [MRG20] specifically for planar regions bounded by line segments and circular arcs. Reinhardt discusses this algorithm under several aspects [Rei23]. Different neighborhoods of a pixel are examined and compared. Reinhardt points out that a neighborhood around a pixel consisting of pixels with initially large but decreasing distances to the pixel leads to fewer iteration steps in the performed experiments. Furthermore, the influence of different texture resolutions and computational precision on the result is investigated. It is indicated that higher precision yields better results. Additionally, a test for local discrete conformality is performed based on discrete conformality. We will perform a similar test as will be presented in Section 5.3.

5.2.1 Pixels inside the stroke

The general concept behind the averaging of preimage pixel coordinates is derived from the local property of conformal maps to be a similarity transformation, i.e., a combination of a translation, a scaling and a rotation (see Definition 1.1.2). Hence, it is necessary that every conformal map f locally satisfies the following: if q is the arithmetic mean of its infinitesimal neighbors, i.e., for $\epsilon > 0$ it holds

$$q = \frac{1}{4} ((q + \epsilon \cdot (1, 0)) + (q + \epsilon \cdot (0, 1)) + (q + \epsilon \cdot (-1, 0)) + (q + \epsilon \cdot (0, -1))),$$

then its image $f(q)$ is approximately given by

$$f(q) \approx \frac{1}{4} (f(q + \epsilon \cdot (1, 0)) + f(q + \epsilon \cdot (0, 1)) + f(q + \epsilon \cdot (-1, 0)) + f(q + \epsilon \cdot (0, -1)))$$

with an error of $o(\epsilon)$ converging to zero if $\epsilon \rightarrow 0$ [Rei23, p.39]. If we visualize springs or rubber bands stretched between $f(q)$ and its neighbors, then iteratively calculating the average of a pixel's neighboring positions leads to a state where all springs are in equilibrium. Hence, repeated calculation of the arithmetic mean for all points in the target domain can be interpreted as the successive convergence to the force equilibrium between all pixels $f(q)$, if it exists. Figure 5.5 displays an instance where the position of $f(q)$ is updated due to the force equilibrium in which it is pulled by the attached springs to its four neighbors.

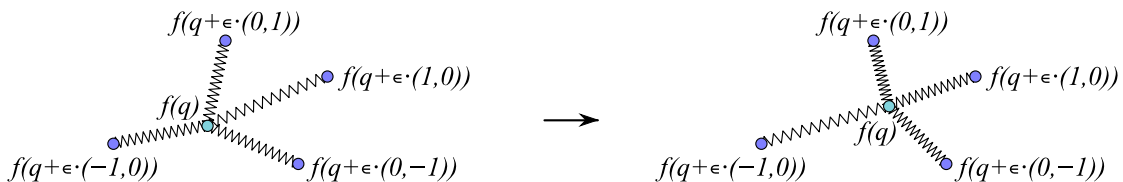


Figure 5.5: Force equilibrium of four springs attached at $f(q)$.

We modify this property and define an iterative algorithm that assigns the average preimage value of its infinitesimal neighbors to all pixels p inside the stroke on the texture `preimageCanvas`. This means that we apply the previous arguments in the inverse direction and search for approximations of the preimage coordinates q under the conformal

inverse f^{-1} of the pixels p in the stroke domain. A reasonable scale for infinitesimal neighborhood on a discrete pixel grid is a distance of at most one or two pixels. We consider up to 16 neighbors of the point p , symmetrically located on two circles with distances of 1 or 2 pixels from p (see Figure 5.6).

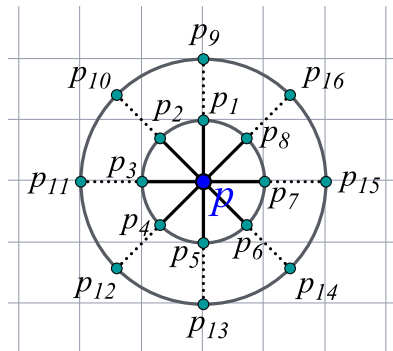


Figure 5.6: The 16 neighbors of a pixel p with a distance of at most 2 pixels.

Pixel p and its 16 neighbors p_1, \dots, p_{16} all lie within the stroke if each of them has a value of 1 in the alpha-channel in texture `centerCanvas`. If this is the case, we can access their currently stored preimage coordinate values in `preimageCanvas` and compute their average coordinate value, which is then stored back in the `rgba`-value of pixel p . See Algorithm 6 for the corresponding CindyJS code.

Algorithm 6 Pixel averaging algorithm for pixels with all 16 neighbors inside the stroke.

```

1: colorplot((0,0),(1024,0), "preimageCanvas",
2: p=#.xy;
3: refVec=imagergba((0,0),(1024,0), "centerCanvas", p, interpolate→false);
4: if(refVec_4 > 0 ,
5:   averagepoint=(0,0); count=0;
6:   repeat(16,i,
7:     refpi=imagergba((0,0),(1024,0), "centerCanvas", pi, interpolate→false);
8:     if(refpi_4 > 0,
9:       count=count+1;
10:    preimVec=imagergba((0,0),(1024,0), "preimageCanvas", pi, interpolate→false);
11:    preimpt=restoreData(preimVec,scvel,base);
12:    averagePoint=averagePoint+(preimpt_1,preimpt_2); ); //end if
13:   if(count==16,
14:     rgba=splitData(averagePoint/count,scvel,base);
15:     , //some boundary handling, see Section 5.2.2); //end if
16:   ); //end repeat
17: , //else if pixel p is outside the stroke
18: rgba=(0,0,0,0); ); //end if
19: rgba );
```

In Algorithm 6, two important details are missing. One is the calculation of the `rgba`-vector in line 15 in the case that not all neighbors p_i of p are situated inside the

stroke. This is the subject of the following Section 5.2.2. The other missing detail is about what happens between lines 11 and 12. In line 11, the preimage coordinates of the neighboring point p_i are obtained from the `preimageCanvas` with the `restoreData` function, and in line 12, these coordinates are added to the sum of all coordinates used for the averaging process. This is not directly applicable without considering that we computed the x -coordinate of $q = \tau_d(p)$ modulo $m \cdot h_T$. It is essential to verify that the preimage coordinates q_i of the pixel p_i for $i \in \{1, \dots, 16\}$ are in their intended positions before using them in the averaging step.

For example, a pixel may lie on the boundary between two consecutive circles, $\mathcal{C}(t_d)$ and $\mathcal{C}(t_{d+1})$. In such cases, neighboring points are located within both circles. It is possible that $\mathcal{C}(t_{d+1})$ is shifted to the beginning of S_m by $\frac{m}{2} \cdot h_T$, while $\mathcal{C}(t_d)$ is located at the end of S_m . Figure 5.7 illustrates this case where, for simplicity, we only consider taking the average over four preimage coordinates of neighbors p_1, p_3, p_5, p_7 of p . We see that only q_7 was shifted and has preimage coordinates q'_7 on the left side of the tile T , while q_1, q_3, q_5 are located on its right side. Directly averaging the coordinates would assign the potential new preimage coordinate $\frac{1}{4}(q_1 + q_3 + q_5 + q'_7)$ to p where it obviously does not belong (see Figure 5.7).

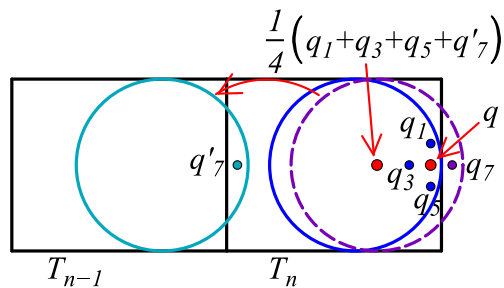


Figure 5.7: Error occurring when modulo calculations are neglected before averaging.

We solve this problem by using the preimage coordinates q_i of neighbors p_i that are located within the same coordinate range in relation to w_T as the current preimage coordinates q of p . To be more precise, if the preimage coordinate $q = \tau_d(p)$ corresponds to a circle with center $c_d = \tau_d(\gamma(t_d))$, then we test whether the preimage coordinates of p 's neighbors p_i for $i \in \{1, \dots, 16\}$ lie within the same preimage circle around c_d with radius $\frac{h_T}{2}$. If this is not the case, they are moved to a position within the circle by a suitable multiple of w_T before the coordinates are used for the averaging. The pseudo code for this adjustment is given in Algorithm 7.

5.2.2 Boundary handling

If fewer than all 16 neighbors of pixel p are contained within the stroke during the pixel averaging Algorithm 6, we know that pixel p is very close to the boundary of the stroke. We say that such a p is in the *boundary area* of stroke s . In this case, we first reduce the number of neighbors considered for the averaging process to eight, i.e., the neighbors p_1, \dots, p_8 with a distance of 1 pixel from p (see Figure 5.6). If all eight neighbors are within the stroke, we perform the usual pixel averaging with a reduced number of neighbors.

Algorithm 7 Adjustment of the preimage coordinates due to the modulo calculations.

for a neighbor p_i of pixel p with $i \in \{1, \dots, 16\}$:

```

 $c_d \leftarrow \tau_d(\gamma(t_d))$  for  $p \in \mathcal{C}(t_d)$ 
 $c_{\bar{d}} \leftarrow \tau_{\bar{d}}(\gamma(t_{\bar{d}}))$  for  $p_i \in \mathcal{C}(t_{\bar{d}})$ 
if  $|c_{\bar{d}} - c_d| > \frac{h_T}{2}$  then
  if the  $x$ -coordinate of  $c_{\bar{d}}$  is bigger than the  $x$ -coordinate of  $c_d$  then
    the  $x$ -coordinate of  $p_i$  is shifted by  $-w_T$ 
  else
    the  $x$ -coordinate of  $p_i$  is shifted by  $+w_T$ 
  end if
else
  the  $x$ -coordinate of  $p_i$  is not shifted
end if

```

If p is located in the boundary area of the stroke s and some of its eight neighbors are outside the stroke, we distinguish two cases (see Definition 2.2.5):

- either p is located near $\mathcal{C}(t_0)$ or $\mathcal{C}(t_N)$, i.e., the first or last circle bounding the stroke,
- p is close to one of the envelope curves w_{\pm} .

For both cases, we have previously encountered the theory of conformal extension. In the first case involving the circular arcs of $\mathcal{C}(t_0)$ and $\mathcal{C}(t_N)$, the Schwarz reflection principle from Theorem 3.2.3 can be applied. As for the envelope curves w_{\pm} , we determined in Proposition 4.1.1 that they are analytic. Thus, we can apply the generalization of the Schwarz reflection principle from Section 3.3.

Schwarz reflection for $\mathcal{C}(t_0)$ and $\mathcal{C}(t_N)$

The Schwarz reflection principle from Theorem 3.2.3 states that a conformal map f from the ornamental strip S_n to the stroke s can be extended along circular arcs α_0 and α_N which are part of the boundary of S_n with $\alpha_0 \subset f^{-1}(\mathcal{C}(t_0))$ and $\alpha_N \subset f^{-1}(\mathcal{C}(t_N))$. It provides an explicit image point for a pixel q^* that is outside the preimage domain (as seen in Figure 5.1). If q is the inverse of q^* with respect to α_0 or α_N , i.e., q is located inside the preimage domain of the stroke, then

$$f(q^*) = \text{refl}_{\mathcal{C}(t_x)}(f(q)) = p^*$$

where $x \in \{0, N\}$ and $\text{refl}_{\mathcal{C}(t_x)}$ is the reflection at the respective circle $\mathcal{C}(t_x)$ of stroke s .

Since we compute the function f^{-1} to find the preimage coordinates for all pixels p within the stroke s , we proceed the other way around. The reverse Schwarz reflection principle can be applied to pixels p in the first or last circle in stroke s that are recognized by the stored number d in the texture `equiCanvas` at pixel p .

Let p^* be one of the eight neighboring pixels p_1, \dots, p_8 of p and let $x \in \{0, N\}$. A pixel p^* is defined as outside of the stroke if the alpha-channel of the `rgba`-vector at p^* vanishes in the texture `centerCanvas`. In this case, the center $\gamma(t_x)$ of the circle $\mathcal{C}(t_x)$ in s is obtained from the texture `centerStrokeCanvas` at p and the corresponding radius $r(t_x)$

from the texture `radiusCanvas`. Then, pixel p^* is reflected at $\mathcal{C}(t_x)$, i.e., $p' = \text{refl}_{\mathcal{C}(t_x)}(p^*)$. The resulting pixel p' is inside the stroke s having the preimage q' assigned to it in `preimageCanvas`, i.e., $q' = \tau_d(p')$. The point q' is reflected back outside the preimage domain of the stroke s within S_m . For this, the center of the corresponding preimage circle c_x of $\mathcal{C}(t_x)$ is read from the texture `centerCanvas` at p' . For $x = 0$, this center is known to be $(\frac{h_t}{2}, \frac{h_t}{2})$, but only the y -coordinate is known for $x = N$. The radius of both preimage circles is $\frac{h_t}{2}$. The preimage of p^* is thus the reflection of q' at the circle with center c_x and radius $\frac{h_t}{2}$. This point, q^* , is then used in the averaging process for pixel p , along with the regular preimage points for neighbors p_i within stroke s , and possibly other pixels treated by Schwarz reflection. See Algorithm 8 for the corresponding pseudocode of this procedure.

Algorithm 8 Schwarz reflection applied to pixels close to $\mathcal{C}(t_0)$ and $\mathcal{C}(t_N)$.

- 1: **if** p^* is not in stroke s **then**
 - 2: reflect p^* at $\mathcal{C}(t_x)$ for $x \in \{0, N\}$: $p' \leftarrow \text{refl}_{\mathcal{C}(t_x)}(p^*)$
 - 3: read the preimage of p' from `preimageCanvas`: $q' \leftarrow \tau_x(p')$
 - 4: reflect q' at the preimage circle of $\mathcal{C}(t_x)$: $q^* \leftarrow \text{refl}_{\tau_x(\mathcal{C}(t_x))}(q')$
 - 5: use q^* for the averaging process of pixel p .
 - 6: **end if**
-

Note 5.2.1 *As before, we need to use the coordinate adjustment from Algorithm 7 when dealing with coordinates computed modulo $m \cdot h_T$. This step is necessary before q' is reflected on the circle c_x , i.e., between lines 3 and 4 in the Algorithm 8. To stabilize the averaging process, another condition is introduced before q' is reflected at the preimage circle c_x : q' is reflected at c_x only if, after the coordinate adjustment, the coordinates of q' are inside c_x . If q' is located outside c_x , the reflection would not result in the desired position of q^* outside the preimage domain. Instead, the averaging would distort the preimage coordinates of p and deform the mapped tile T within the stroke.*

Adapted generalization for analytic envelope curves w_{\pm}

If the pixel p is not within the boundary area of the first or last circle of the stroke s , it is on or very close to the envelope curves w_{\pm} . As shown in Proposition 4.1.1, the envelope curves are analytic functions that are C^1 -continuous due to the C^2 -continuous cubic B-spline interpolation of the curve γ and the radius function r . We showed in Section 3.3 that the Schwarz reflection principle can be generalized for analytic curves. Rather than reflecting at circular arcs or line segments, the reflection can be generalized at an analytic curve to local reflection across the tangent at the closest curve point. Hence, the conformal map f from the preimage domain in the tiled strip S_n to the stroke s can be extended to include points q^* outside the preimage domain by

$$f(q^*) = \text{refl}_{w'_{\pm}(t_x)}(f(q)) = p^*$$

with q being the reflection of q^* across the line $y = h_T$ or $y = 0$ depending on the position of q^* . The vector $w'_{\pm}(t_x)$ is the directional vector of the tangent $\mathcal{T}_{\pm}(\lambda) = w_{\pm}(t_x) + \lambda w'_{\pm}(t_x)$

at the nearest boundary point $w_{\pm}(t_x)$ to $f(q)$ on the corresponding envelope curve. Specifically, $\mathcal{T}_+(\lambda)$ is used for reflecting q^* at $y = h_T$, and $\mathcal{T}_-(\lambda)$ for a reflection at $y = 0$. Again, our focus is on the inverse process of determining a preimage point q^* for a given pixel p^* outside of stroke s . However, despite the knowledge about the generalized Schwarz reflection principle, this is not a straightforward task.

Let pixel p^* outside stroke s be one of the eight neighboring pixels p_1, \dots, p_8 of pixel p that are not in the boundary regions of the first or last circles $\mathcal{C}(t_0)$ and $\mathcal{C}(t_N)$. The first step in the inverse procedure of the generalized Schwarz reflection principle would be to reflect p^* at $\mathcal{T}_{\pm}(\lambda)$. It can be easily determined whether the correct tangent is $\mathcal{T}_+(\lambda)$ or $\mathcal{T}_-(\lambda)$ by testing whether the preimage of p is close to $y = 0$ or to $y = h_T$ in S_m . However, computing the nearest point on the corresponding envelope curve is challenging despite having the concrete formulas for the piecewise defined envelope curves w_{\pm}^d as in Equation (4.1). Besides, the equations of the envelope curves necessary for the calculation of the tangent as well as the discrete data points $P_{t_{d-1}}, P_{t_d}, P_{t_{d+1}}, P_{t_{d+2}}$ or the derivatives of the envelope curves w'_{\pm} as given in Equation (2.14) are not stored. Especially if the data is to be accurate enough to be used for exact calculations, it would be necessary to create a large amount of additional textures to store all this information. Experiments have shown that this would significantly slow down the calculations.

Instead of reflecting p^* at the nearest tangent $\mathcal{T}_{\pm}(\lambda)$ of the envelope curve, we use the information already stored in the textures specified in Section 5.1.2. Nevertheless, we use that the generalization of Schwarz's reflection principle states that locally, the reflection at the analytic curve w_{\pm} is the same as the reflection at its tangent. Our analytic curve is defined as the envelope curve of a family of circles. This means that the envelope curves w_{\pm} are tangent to the respective circles by definition. We already store the circle center $\gamma(t_d)$ of the circle $\mathcal{C}(t_d)$ to which the pixel p belongs in `centerStrokeCanvas` and the corresponding radius $r(t_d)$ in `radiusCanvas`. With the given information, we can reflect p^* directly at the circle $\mathcal{C}(t_d)$, since locally this reflection is the same as the reflection at the circle's tangent for a point very close to the circle, as it is the case for p^* .

However, we are not dealing with a continuous stroke but with a discrete set of circles. This is why, depending on the distance between the selected curve points $\gamma(t_d)$ in Algorithm 1, the actual boundary of the registered stroke s on the textures has a slightly wavy character, as shown in Figure 4.5 on the right, but which is not visible to the human eye. Since our algorithm assigns information from new circles to pixels within the previously uncovered circular crescent of the continuously growing stroke, as outlined in Section 5.1.2, there is a visible effect when p^* is reflected at the assigned circle $\mathcal{C}(t_d)$. The direction of reflection at circle $\mathcal{C}(t_d)$ usually is not perpendicular to the (piecewise) continuously computed tangent to the envelope curves w_{\pm} . This can be observed in Figure 5.8 on the right, where we closely examine the borders of a stroke and analyze three nearby pixels (p_j^* for $j \in \{1, 2, 3\}$) outside the stroke. When the preimage of the reflected point is itself reflected out of the strip to the preimage points q_j^* , as it is depicted on the left side of Figure 5.8, the direction of the reflection at the circle in the stroke is not compensated. As a result, the artistic content of the tiled strip seems to be pulled towards the opposite direction. A static example is given in Figure 5.9, where the left picture was taken a few seconds before the right one. The blue line through the stroke is a visual marker to better see the difference between the ornaments in the two strokes.

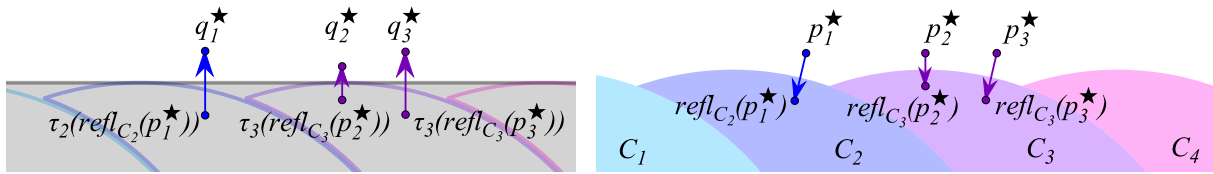


Figure 5.8: Reflection at circles instead of tangents to the envelope curve.

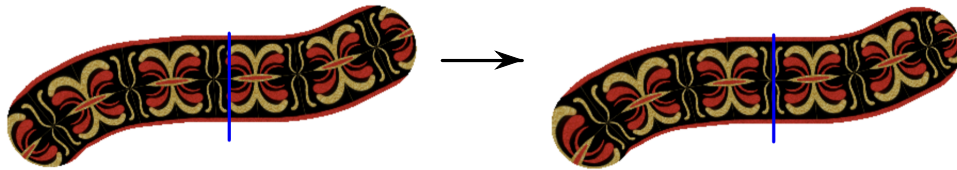


Figure 5.9: The ornament flowing to the right while averaging is active.

All in all, we adjust the generalized Schwarz reflection once more and modify the reflection step at the stroke boundary. The following steps describe the procedure for handling neighboring pixels p_i for $i \in \{1, \dots, 8\}$ of pixel p that are outside the stroke but not in the boundary area close to the first or last circle of stroke s . Figure 5.10 gives a visualization.

- 1) We determine whether the opposite neighbor p'_i of p_i , i.e., p_i rotated around p by 180° , is inside the stroke. The pairs of opposite neighbors are p_1p_5 , p_2p_6 , p_3p_7 , and p_4p_8 , as in Figure 5.6.
- 2) If p'_i is located within the stroke, its corresponding preimage coordinate q'_i is read from the texture `preimageCanvas`. Otherwise, both adjacent pixels p_i and p'_i are not included in the averaging process
- 3) The preimage coordinate q'_i is reflected at the upper or lower boundary of the rectangular strip, i.e., at $y = 0$ or $y = h_T$ depending on the y -coordinate of q'_i .
- 4) This reflected point q_i is included in the averaging calculation of the preimage coordinates for pixel p .

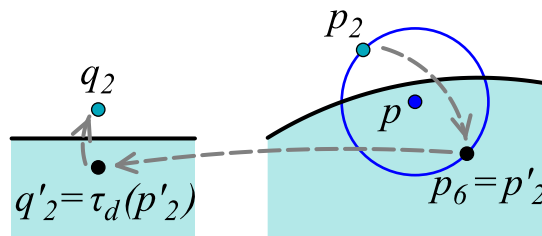


Figure 5.10: Adapted “reflection” for pixels outside the stroke near an envelope curve.

With this method to determine preimage points for pixels outside the stroke near w_\pm , we have completed the missing cases of the pixel averaging Algorithm 6 in line 15. Thus,

we can proceed to the next questions: does pixel averaging yield a nearly conformal solution? Moreover, does the averaging process converge? After how many iterations do we stop the averaging process? So far, we have only specified that the computations are performed in real time, simultaneously to the drawing process. However, if the averaging stopped when the user stops drawing, the newer part of the stroke would have been treated with fewer iterations of the averaging process than the older part. Figure 5.11 shows a stroke where the averaging was interrupted when the drawing stopped. The pixel averaging algorithm did not (sufficiently) process the right end of the stroke. Therefore, it is necessary to continue with the averaging process after the stroke is completed, and we need to define some rules for when the averaging has to stop.



Figure 5.11: Interruption of pixel averaging when the drawing stops.

5.3 Test for conformality and stopping criterion

To validate the accuracy of our algorithm and establish a stopping criterion for the pixel averaging algorithm outlined in the previous section, it is necessary to define a reference measure of the distortion of the coordinate system in the stroke domain. For this, we will propose a method based on the properties of discrete conformal maps discussed in Section 4.2.1, which locally measures for each pixel in stroke s how close the map is to being conformal. Additionally, we will introduce a global energy that will serve as a stopping criterion for the averaging process once a local minimum of this energy is reached. We will use these criteria to compare the energy values resulting from strokes treated with our averaging process to those resulting from strokes treated without or with limited averaging. Furthermore, we will compare our algorithm's output for the stroke of Benchmark Example 2.1.7 in the form of a half annulus with an exact and continuous conformal map from a finite rectangular strip to a half annulus with the same parameters. All measures of accuracy, both visual and numeric, indicate that our maps are indeed nearly conformal.

5.3.1 Pixel based test for conformality

The stroke and its corresponding preimage domain in the tiled strip S_n are saved as discrete textures consisting of a finite rectangular grid of pixels. Locally, around each interior pixel p_0 , the grid looks like the one in Figure 5.12 on the left. The color coding in both pictures in Figure 5.12 will be explained below.

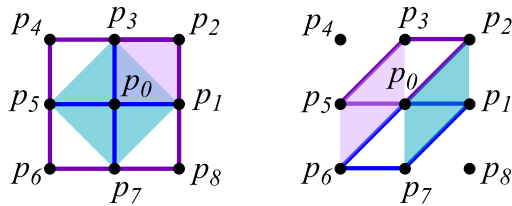


Figure 5.12: Rectangular grid of pixels around p_0 in the stroke.

In Section 4.2.1, we have examined the criteria for discrete conformal equivalence of two combinatorially equivalent polygonal surfaces. Proposition 4.2.3 states that two cyclic quadrangulations are discretely conformally equivalent if and only if for each face the length-multi-ratios are equal. The pixels of the stroke texture are arranged in squares, so they are cyclic, and we can apply Proposition 4.2.3 to test whether the preimage coordinates stored in texture `preimageCanvas` satisfy the same length-multi-ratios as the squares in the stroke. Note that the quadrangulations are combinatorially equivalent by definition, since the vertices in the preimage domain are defined as the preimages of the pixels within the stroke.

For each pixel p_0 within the stroke, we consider its neighbors $p_1 = p_0 + (1, 0)$, $p_2 = p_0 + (1, 1)$ and $p_3 = p_0 + (0, 1)$, forming a polygon within the grid (see the light purple square in Figure 5.12 on the left). Their length-multi-ratio (`lmr`) is given by

$$\text{lmr}_{p_0 p_1 p_2 p_3} = \frac{\ell_{p_0 p_1} \ell_{p_2 p_3}}{\ell_{p_1 p_2} \ell_{p_3 p_0}} = \frac{|p_0 - p_1| \cdot |p_2 - p_3|}{|p_1 - p_2| \cdot |p_3 - p_0|} = \frac{1 \cdot 1}{1 \cdot 1} = 1.$$

To determine whether the calculated map between the strip and the stroke is (approximately) discretely conformal, we compute the length-multi-ratio of the preimage coordinates q_0, q_1, q_2, q_3 of the pixels p_0, p_1, p_2, p_3 read from `preimageCanvas`

$$\text{lmr}_{q_0 q_1 q_2 q_3} = \frac{\ell_{q_0 q_1} \ell_{q_2 q_3}}{\ell_{q_1 q_2} \ell_{q_3 q_0}}$$

and compare it to $\text{lmr}_{p_0 p_1 p_2 p_3} = 1$. If it holds that $|\text{lmr}_{q_0 q_1 q_2 q_3} - 1| \approx 0$, then the two quadrangulations of the stroke and its preimage domain are (approximately) discretely conformally equivalent.

For two combinatorially equivalent triangulations, Proposition 4.2.2 states that they are discretely conformally equivalent if and only if the length-cross-ratios along all interior edges are equal. To apply this local criterion, it is necessary to extend the rectangular pixel grid to a triangulation, which is achieved by adding one diagonal per pixel square. We include the diagonals that link the bottom left and top right pixels in each pixel square (see the diagonals in Figure 5.12 on the right).

Hence, we examine for each pixel p_0 and all its outgoing edges ij whether the length-cross-ratio of the adjacent triangles along ij is preserved by the computed map. More precisely, we test four of the outgoing edges from p_0 , namely those to p_1 (consider the adjacent filled blue quadrilateral on the right in Figure 5.12), to p_3 (see the adjacent

purple bordered quadrilateral), to p_5 (see the filled light purple quadrilateral), and to p_7 (consider the blue bordered quadrilateral):

$$\begin{aligned} \text{lcr}_{p_0p_1} &= \frac{\ell_{p_0p_2}\ell_{p_1p_7}}{\ell_{p_0p_7}\ell_{p_1p_2}} = 2 \stackrel{?}{\approx} \text{lcr}_{q_0q_1} & \text{lcr}_{p_0p_3} &= \frac{\ell_{p_0p_2}\ell_{p_3p_5}}{\ell_{p_0p_5}\ell_{p_2p_3}} = 2 \stackrel{?}{\approx} \text{lcr}_{q_0q_3} \\ \text{lcr}_{p_0p_5} &= \frac{\ell_{p_0p_6}\ell_{p_3p_5}}{\ell_{p_0p_3}\ell_{p_5p_6}} = 2 \stackrel{?}{\approx} \text{lcr}_{q_0q_5} & \text{lcr}_{p_0p_7} &= \frac{\ell_{p_0p_6}\ell_{p_1p_7}}{\ell_{p_0p_1}\ell_{p_6p_7}} = 2 \stackrel{?}{\approx} \text{lcr}_{q_0q_7} \end{aligned} \quad (5.5)$$

The other two edges connecting p_0 and p_2 as well as p_0 and p_6 are omitted here, since their adjacent quadrilaterals were already covered by the previous length-multi-ratio test, and since all our tests for local discrete conformality will be combined later.

To also include the property of conformality that angles are preserved locally, we examine whether right angles are preserved by considering the intersection of the straight line segments p_1p_5 and p_3p_7 crossing at pixel p_0 . The two vectors $v_1 = p_1 - p_5$ and $v_2 = p_3 - p_7$ are perpendicular to each other by construction, i.e., their scalar product vanishes:

$$\frac{\langle v_1, v_2 \rangle}{|v_1| \cdot |v_2|} = 0.$$

We examine whether the respective vectors for the preimage coordinates q_1, q_3, q_5, q_7 also enclose a right angle in the strip:

$$\frac{\langle q_1 - q_5, q_3 - q_7 \rangle}{|q_1 - q_5| \cdot |q_3 - q_7|} \stackrel{?}{\approx} 0.$$

Moreover, we refer to the results of Palka [Pal75] discussed in Section 3.2.1, which state that for two regions with the same conformal modulus, or for the image region having a smaller modulus than the preimage region, a conformal map is given if all squares and all rectangles with sides parallel to the coordinate axes preserve the conformal modulus. As previously discussed, we will keep the computations local, which prevents us from checking *all* moduli of squares and rectangles inside the stroke. But we have already considered for each local square $p_0p_1p_2p_3$ whether its modulus 1 is also fulfilled by its preimage quadrilateral $q_0q_1q_2q_3$. Additionally, we include two more squares with a length-cross-ratio of 1 around pixel p_0 , namely $p_1p_3p_5p_7$ shown as the filled blue square in Figure 5.12 on the left and $p_2p_4p_6p_8$ which is the purple bordered square.

Note 5.3.1 *Although we limit the result of Palka [Pal75] to local tests around p_0 , we could include more rectangles with edges parallel to the coordinate axes to our test. For example, we could include the four rectangles $p_1p_2p_4p_5$, $p_1p_5p_6p_8$, $p_2p_3p_7p_8$, and $p_3p_4p_6p_7$. Interestingly, our experiments show that all the ratios and angle conditions considered so far give very good results for our algorithm, while the test of the previous four rectangles deviates from these good results. Since we have yet to determine the reason behind this, we acknowledge the fact but we will not include these rectangles in our tests hereafter.*

As mentioned before, we combine the above tests and calculate a deviation value for each pixel p_0 in stroke s that has eight neighbors inside the stroke. Again, it is necessary to adjust the coordinates of the preimage due to modulo calculations before calculating

the ratios and the angle. After calculating all test values, we equally weight the test for angle preservation, the test for the length-cross-ratios of the quadrilaterals in Figure 5.12 to the right (i.e., Equations (5.5)), and the test for all squares around p_0 (as shown in Figure 5.12 to the left). The resulting total deviation δ_{p_0} is the value for each pixel p_0 that we proceed with:

$$\frac{1}{3} \left(\left| \frac{\langle q_1 - q_5, q_3 - q_7 \rangle}{|q_1 - q_5| \cdot |q_3 - q_7|} \right| + \frac{1}{4} (|\text{lcr}_{q_0q_1} - 2| + |\text{lcr}_{q_0q_3} - 2| + |\text{lcr}_{q_0q_5} - 2| + |\text{lcr}_{q_0q_7} - 2|) \right. \\ \left. + \frac{1}{3} (|\text{lmr}_{q_0q_1q_2q_3} - 1| + |\text{lmr}_{q_1q_3q_5q_7} - 1| + |\text{lmr}_{q_2q_4q_6q_8} - 2|) \right) = \delta_{p_0} \quad (5.6)$$

As previously mentioned, our pixel averaging method operates in real time during and after the drawing process. The flow of coordinates to their positions is visible due to the displayed texture `currentCanvas`, which reads the respective colors from the ornamental tile at the current preimage points for all pixels throughout the averaging process. To provide a visual indicator of the accuracy of the computed map, we introduce an additional texture called `conformalTestCanvas`, on which the deviation values δ_{p_0} are stored for each pixel inside the stroke. To be precise, the deviation value is converted into a color value with the CindyJS command `hue`, which converts a value between zero and one into an *rgb*-color-vector. Since the `hue` function produces a green color at $\text{hue}(1/3)$, and since we want the color to be green when the result of δ_{p_0} , i.e., the local deviation from conformality, is close to zero, we add the value of $1/3$ to δ_{p_0} . Overall, $\text{hue}(1/3 + \delta_{p_0})$ is stored in the red-, green- and blue-channels of `conformalTestCanvas` at pixel p_0 . If a visual test for conformality of the current averaging calculations is desired, the alpha-channel is set to some value greater than zero and the texture `conformalTestCanvas` is drawn additionally to `currentCanvas`, which contains the ornament colors.

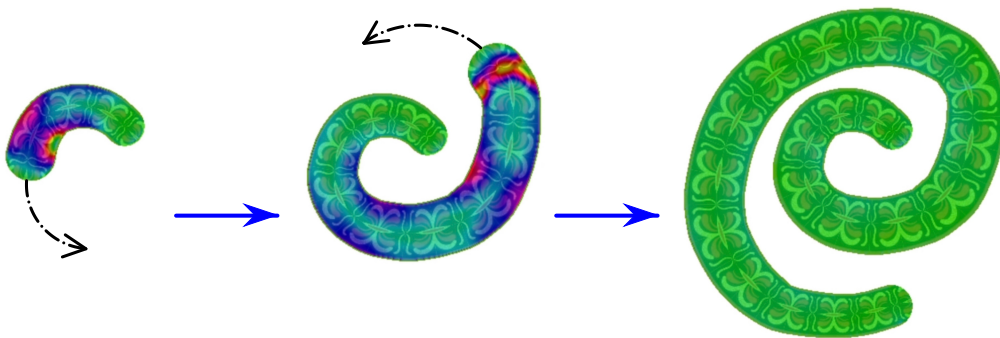


Figure 5.13: Test for conformality while the stroke is drawn.

Figure 5.13 shows an example of how the color in the texture `conformalTestCanvas` changes over time as a stroke is drawn and pixel averaging is applied. The green color indicates that the locally calculated deviation from conformality is close to zero. While drawing the stroke from left to right, certain areas of the stroke display red, yellow, or blue colors (visible in the two stroke parts on the left and in the middle), which shows

that the averaging algorithm seems to gradually converge to good result as it runs (see the final stroke on the right). To validate this accuracy and to provide a stopping criterion for the averaging procedure, we will use the given local deviation values δ_{p_0} to define a global measure of the map's deviation from conformality in the next section.

5.3.2 Local minimum of average error

With the local deviation measure δ_{p_0} for each pixel p_0 in the stroke, we define a global criterion besides the visual one to determine when a stroke is close to conformal and when the pixel averaging needs to be stopped. The fundamental concept of this global criterion is that the deviation δ_p of all pixels p in stroke s is summed and divided by the number of pixels in s . This mean deviation is monitored in each averaging iteration after the completion of the stroke. If the deviation decreases or remains stable in two consecutive steps, the averaging continues. If the deviation increases, the process is stopped. If the criteria did not terminate the averaging iterations within a specific time after completing the stroke, the algorithm automatically stops averaging. Otherwise it may continue indefinitely if the algorithm finds a global minimum, for example.

Again, we calculate the global criterion for conformality by using the fast and local computations on textures. For this, a new texture `globalTestCanvas` is created, which has the same dimensions as the other textures, i.e., 1024×512 pixels. Initially, every pixel within the stroke is assigned an *rgba*-vector of $(\delta_p, 0, 0, 1)$, representing the deviation δ_p from Equation (5.6). Each pixel outside the stroke is assigned a zero *rgba*-vector of $(0, 0, 0, 0)$. Then Algorithm 9 is applied, which sums over all pixels in `globalTestCanvas`. This algorithm was developed based on an algorithm by Werner [Wer23].

Algorithm 9 Calculation of the global average deviation from local conformality.

```

1: globalTestCanvas  $\leftarrow$   $(\delta_p, 0, 0, 1)$  at every pixel  $p$  inside the stroke
2: for  $n = 0 \dots 8$  do
3:    $\forall$  pixels  $p = (x, y)$  in globalTestCanvas:
4:      $(r, g, b, a)_p \leftarrow (r, g, b, a)_p + (r, g, b, a)_{(x+2^n, y)} + (r, g, b, a)_{(x, y+2^n)} + (r, g, b, a)_{(x+2^n, y+2^n)}$ 
5: end for
6: totalAmount  $\leftarrow (r, g, b, a)_{(0.5, 0.5)} + (r, g, b, a)_{(512.5, 0.5)}$ 
7: averageGlobalError  $\leftarrow \frac{1}{\text{totalAmount}_a} \cdot \text{totalAmount}_r$ 

```

Proposition 5.3.2 *After applying Algorithm 9, the variable `averageGlobalError` contains the average deviation from local conformality over all pixels up to the uncertainty in which order the GPU computes the *rgba*-vectors in line 4.*

Proof: The texture `globalTestCanvas` contains 1024×512 pixels which are centered at $(a + 0.5, b + 0.5)$ for $a, b \in \mathbb{N}_0$ (see Section 5.1.3). Every pixel p contains either the vector $(0, 0, 0, 0)$ if it is outside the stroke s , or the vector $(\delta_p, 0, 0, 1)$ if p lies within the stroke s .

The loop iterates for $n = 0$ as long as $n < 9$. After the initial iteration with $n = 0$, the vector $(r, g, b, a)_{(0.5, 0.5)}$ contains the sum of all pixel values in the 2×2 square having $(0.5, 0.5)$ in the lower left corner, i.e., it contains the values of the pixels located at

$(0.5, 0.5)$, $(1.5, 0.5)$, $(0.5, 1.5)$, and $(1.5, 1.5)$. This holds for every pixel, up to the uncertainty of the order in which the GPU performs the computations. It also holds for $p \in \{(0.5, 2.5), (2.5, 0.5), (2.5, 2.5)\} = \{(0.5, 0.5 + 2^1), (0.5 + 2^1, 0.5), (0.5 + 2^1, 0.5 + 2^1)\}$ that they contain the values of the 2×2 square to their top right. It may happen that $(r, g, b, a)_p$ already contains the values of the 2×2 square of which it is the lower left corner when it is added to the *rgba*-value of another pixel. Since this is not controllable by the algorithm, we neglect this uncertainty in the following.

Assume that after the iteration step with $n \in \{1, \dots, 7\}$ the vectors $(r, g, b, a)_{(0.5, 0.5)}$, $(r, g, b, a)_{(2^{n+1}+0.5, 0.5)}$, $(r, g, b, a)_{(0.5, 2^{n+1}+0.5)}$, and $(r, g, b, a)_{(2^{n+1}+0.5, 2^{n+1}+0.5)}$ contain the sum of all pixel values within a $2^{n+1} \times 2^{n+1}$ square with the corresponding pixel as the lower left corner. Then, in the next step for $n \rightarrow n + 1 = n'$, the sum over $(r, g, b, a)_{(0.5, 0.5)}$, $(r, g, b, a)_{(2^{n'}+0.5, 0.5)}$, $(r, g, b, a)_{(0.5, 2^{n'}+0.5)}$, and $(r, g, b, a)_{(2^{n'}+0.5, 2^{n'}+0.5)}$ is stored to pixel $(0.5, 0.5)$. This sum contains all values of the square of $2^{n+1} \times 2^{n+1}$ pixels with $(0.5, 0.5)$ as its lower left corner. As a result, the pixel at $(0.5, 0.5)$ contains the sum over all pixel values within the left $512 \times 512 = 2^9 \times 2^9$ square of `globalTestCanvas` in the last iteration for $n = 8$. The same reasoning applies to the pixel $(512.5, 0.5)$, which contains the sum over all pixel values of the right square in the rectangular texture.

The two *rgba*-vectors $(r, g, b, a)_{(0.5, 0.5)}$ and $(r, g, b, a)_{(512.5, 0.5)}$ are added in line 6, resulting in the sum over all entries of the original texture `globalTestCanvas`:

$$\text{totalAmount} = \left(\sum_{p \in s} \delta_p, 0, 0, \sum_{p \in s} 1 \right).$$

The alpha-channel of `totalAmount` indicates the number of pixels p in stroke s since all pixels outside the stroke in texture `globalTestCanvas` initially contained a zero in their alpha-channel. The first entry contains the sum of all deviation values of pixels in the stroke. The average global error is thus calculated as the sum of all deviation values divided by the number of pixels that contributed to that deviation, as in line 7 of Algorithm 9. \square

If the algorithm counted the deviation value of some pixels in the red-channel of `totalAmount` more than once due to the uncertainty in which order the GPU executes the calculations, then the alpha-channel also increases by the amount of multiply counted deviations. Hence, the uncertainty in which order the GPU computes the sums leads to a situation where some pixels of the stroke affect the global deviation from conformality more than others.

With the result from Proposition 5.3.2 and Algorithm 9, we have a global average error that distinguishes a perfectly conformal map of the tiled strip S_n to the stroke from the map computed by the pixel averaging procedure in Algorithm 6 together with the boundary handling from Section 5.2.2. As already stated, we use this global average error to validate our method and to define a stopping criterion for the pixel averaging. Once the user stops drawing, the absolute value of the global average error over the entire stroke domain is tested if it decreases or stays the same. In this case, further averaging is applied to all pixels belonging to the stroke domain. If the global error increases, this indicates that the averaging has found a local minimum of the global error and the averaging procedure is stopped. Alternatively, the averaging process is stopped after some seconds,

since it may happen that the algorithm has found a stable minimum of the global error. In both cases, the result is an approximation of a conformal map of the tiled ornamental strip to the stroke. Some examples are shown in Figure 5.14 along with the original ornamental tiles. The second example demonstrates that the circles in the tile are preserved by the algorithm, i.e., they are still circles after the application of our mapping algorithm.

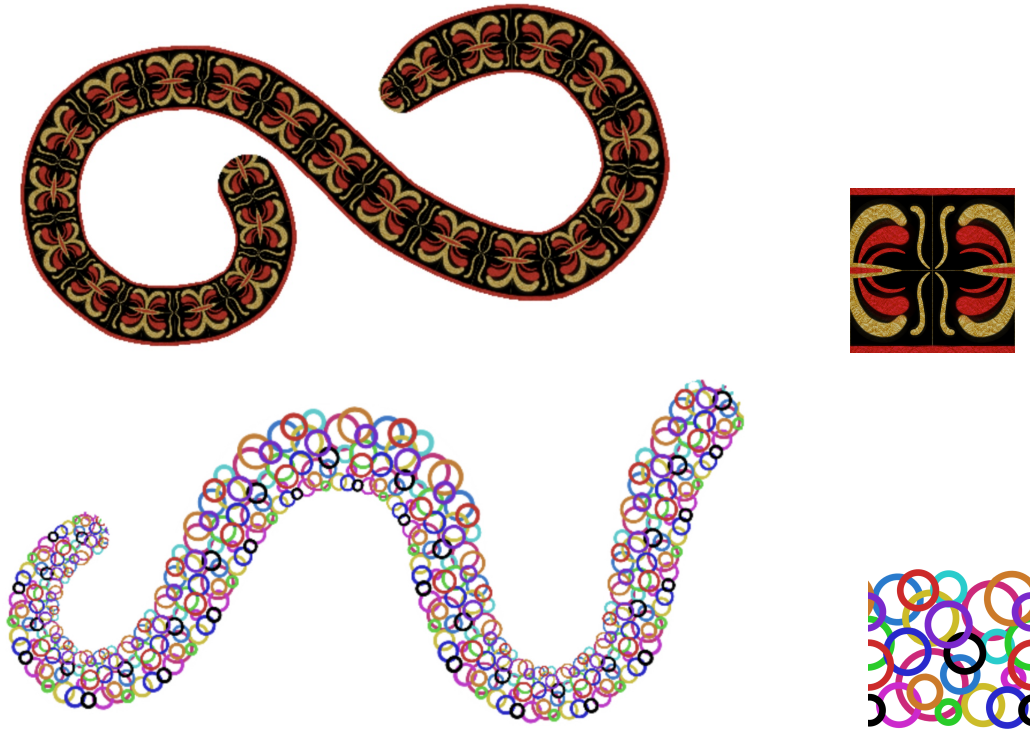


Figure 5.14: Results of the pixel averaging algorithm.

The final value of `averageGlobalError` provides a numerical measure of the quality of the approximation of a conformal map by our pixel averaging algorithm. Table 5.1 presents some values that classify the accuracy of our algorithm. It lists the values of `averageGlobalError` for Benchmark Example 2.1.7 and a stroke drawn freely in the following cases:

- 1) Only the initial coordinates for all pixels in stroke s are calculated and stored in `preimageCanvas`, but no averaging is performed.
- 2) Pixel averaging is applied only to pixels located inside the stroke, while those in the boundary area with less than 16 neighbors inside the stroke are not considered.
- 3) Full pixel averaging, including boundary handling as described in Section 5.2.2, is employed until the user stops drawing.
- 4) The algorithm is fully applied until the stopping criterion terminates the averaging process.

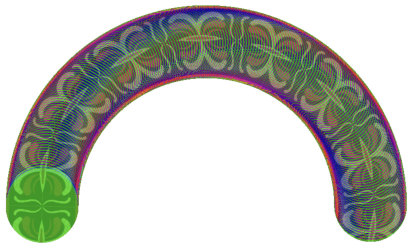
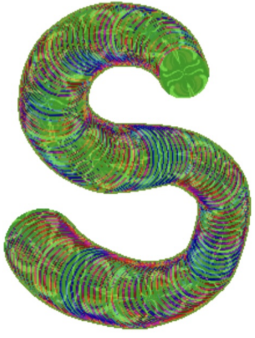


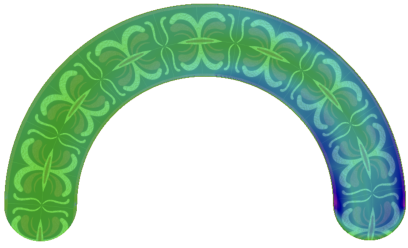

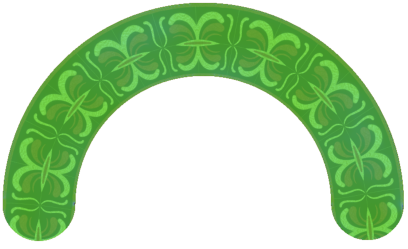

		Half annulus		Stroke example
1)	0.2509		0.4871	
2)	0.0614		0.1396	
3)	0.1042		0.2033	
4)	0.0514		0.0452	

Table 5.1: Average global errors of the corresponding strokes.

As expected, the lowest total deviation is achieved by the strokes in case 4), where neither the pixel averaging is stopped earlier nor the boundary handling of the pixel averaging is omitted.

5.3.3 Comparison of conformal moduli

As announced in Section 3.2.2, we will compare the modulus of the explicit conformal map of a half annulus A with the modulus of the rectangular ornamental strip S_n , which our algorithm mapped onto the half annulus stroke s_A of the Benchmark Example 2.1.7. If the conformal modulus of the annulus A is significantly different from the modulus of the strip S_n , the exact calculations of the conformal map $F : R \rightarrow A$ probably also differ significantly from the result of our algorithm mapping S_n to s_A .

As detailed in Section 3.2.2, the half annulus A consists of all points $z \in \mathbb{C}$ with $\rho < |z| < \frac{1}{\rho}$ and $\text{Im}(z) > 0$ for a parameter $\rho \in (0, 1)$. The rectangle R consists of points $w = x + iy$ with $x \in (-\frac{\pi}{2}, \frac{\pi}{2})$ and $y \in (0, 2 \ln(\rho))$. It is mapped to A by $f(w) = i\rho e^{-iw}$. Therefore, the unique conformal modulus of the annulus A is given by the side-length-ratio of the rectangle R as $M(R) = M(A) = -\frac{\pi}{2 \ln(\rho)}$ according to Equation (3.5).

The stroke s_A that most closely matches A is given by $\gamma(t) = \frac{\frac{1}{\rho} - \rho}{2} (\cos(\pi - t), \sin(\pi - t))$ for $t \in [0, \pi]$ and with fixed radius $r_c = \frac{\frac{1}{\rho} - \rho}{2}$. Figure 3.8 shows a visualization of the half annulus A and the stroke s_A side by side. The side-length-ratio of the rectangular strip that corresponds to s_A provides an estimate of the conformal modulus of s_A .

We use the preimage tile of height h_T and width w_T to determine the side lengths of the rectangular preimage of stroke s_A . In Equation (5.2), the location of the new preimage point of the center $\tau_d(\gamma(t_d))$ is calculated for each new circle $\mathcal{C}(t_d)$ on the horizontal line $y = \frac{h_T}{2}$ within the rectangular strip S_n . Since the preimage center of the first circle is located at $(\frac{h_T}{2}, \frac{h_T}{2})$ and since that of the last circle of the stroke $\mathcal{C}(t_N)$ is located at $(x_N, \frac{h_T}{2})$, the total width of S_n equals $x_N + \frac{h_T}{2}$ when the first and last half circles are included (see Figure 5.1). The length-ratio of the rectangular strip S_n which our algorithm maps to the stroke s_A is thus given by

$$M(S_n) = \frac{1}{h_T} \left(x_N + \frac{h_T}{2} \right).$$

To better compare stroke s_A with half annulus A , it is necessary to exclude the first and last half circles in s_A . The value to which we compare the conformal modulus of the half annulus A is thus given as

$$M(S_n) - 2 \cdot \frac{h_T}{2} = \frac{1}{h_T} \left(x_N - \frac{h_T}{2} \right).$$

Since the half circles of $\mathcal{C}(t_0)$ and $\mathcal{C}(t_N)$, as well as their corresponding preimage half circles around $\tau_0(\gamma(t_0))$ and $\tau_N(\gamma(t_N))$, were not considered in the calculation of the approximate conformal map between S_n and the stroke s_A , any comparison is only an approximation.

Table 5.2 lists the modulus of the half annulus A and R for different parameters ρ , as well as the corresponding length-ratio of the strip S_n associated with the stroke s_A for the same parameters. The results indicate that there is no significant difference in the length-ratios between the rectangles R and the rectangular strips S_n . Hence, this again serves as a confirmation that our pixel averaging algorithm does indeed compute valid approximations of conformal maps.

parameter ρ	half annulus A	rectangular preimage strip	\approx difference
0.97	51.5705	51.4727	0.1
0.95	30.6238	30.5831	0.04
0.9	14.9088	14.9309	-0.02
0.85	9.6653	9.7288	-0.06
0.8	7.0394	7.1401	-0.1

Table 5.2: Conformal modulus of several half annuli and the corresponding strokes.

5.4 Adaptions for non-univalent strokes

In previous sections, we have introduced our algorithm for computing conformal maps between an ornamental strip and a user-drawn stroke. Up to now, we have only analyzed univalent strokes without self-intersection and singular boundary points. In this section, we will present adaptations made to the previously presented algorithm in case the stroke splits into two parts due to self-intersection. Furthermore, we will provide a concept on how to modify the algorithm for cases where the boundary of the stroke contains singular points, i.e., cusps.

5.4.1 Extension of the algorithm for self-intersections

To incorporate the necessary changes required to cover self-intersecting strokes with our algorithm, we recall that Algorithm 2 from Section 4.3.1 enables us to detect whether a stroke is self-intersecting or not. This detection of self-intersection is integrated into the global algorithm after the repeated calculation of equidistant circle centers $\gamma(t_d)$ on the interpolated curve γ . When the new circular crescent of circle $\mathcal{C}(t_{d^*})$ causes self-intersection of the stroke, the algorithm must be adapted so that it is able

- to cover parts of the drawing surface more than once,
- to apply pixel averaging to the preimage coordinates, even in cases of self-intersection,
- to test the union of all stroke parts for conformality.

For this, we define new textures on which the stroke continues from the moment of self-intersection, i.e., for all $d \geq d^*$. This is necessary because if the circular crescent $\mathcal{C}(t_{d^*}) \setminus \mathcal{C}(t_{d^*-1})$ intersects with the current stroke s , storing information for pixels in that crescent would overwrite the *rgba*-vectors of pixels in s . Hence, for all textures listed in Section 5.1.2, second versions are created which are used for the second part of the stroke. Figure 5.15 gives a visualization of the two parts of a self-intersecting stroke: the red circle on the left causes self-intersection with the exiting stroke, which is why the second texture contains all new circular crescents filled in blue from that moment on. Combining the two textures, the entire stroke is shown on the right. Besides the red circle $\mathcal{C}(t_{d^*})$, the orange circle $\mathcal{C}(t_{d^*-1})$ will also become important in the following.

In addition to the textures listed in Section 5.1.2, a second texture `currentCanvas2` is created to display the results of the reverse pixel lookup. The reverse pixel lookup computes the preimage coordinates as before, the results are just stored on a different

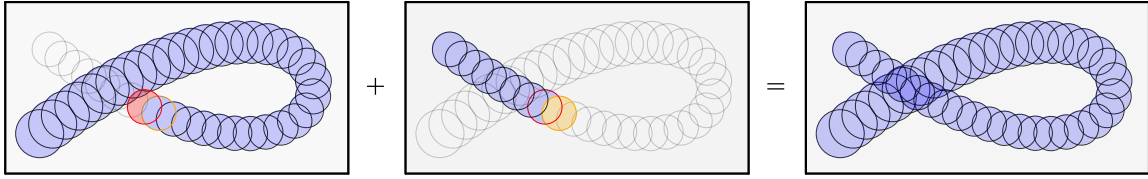


Figure 5.15: Textures containing the two parts of a self-intersecting stroke.

texture, and the color for pixel p on `currentCanvas2` is still read from the same tiled strip S_m .

So far, this has only been implemented for a single self-intersection. However, this method can be applied as many times as necessary to allow for multiple self-intersections of a single stroke. Note that the second part of a stroke may intersect the first part multiple times, as the second textures contain no preimage coordinates or other stored information for all pixels of the previous part of the stroke. A third version of all textures is only required when the second part of the stroke intersects itself, and so on.

When pixel averaging is applied, it is necessary to allow interaction between the first and second versions of the textures. Since the stroke is divided into two parts, it is important to consider that the circle crescent $\mathcal{C}(t_{d^*})$ and the circle $\mathcal{C}(t_{d^*-1})$ contain neighboring pixels. Hence, in order to see whether a neighboring pixel p_i of a pixel p is part of the stroke or not, it is not enough to check for a (non-)vanishing alpha-channel of this neighboring pixel p_i in the textures `centerCanvas(2)` to which the pixel p belongs. Additionally, examining nearby pixels in the respective other texture would cause the circles of the second part of the stroke that cover the first part of the stroke to be incorrectly adjacent to the underlying circles of the first part.

To solve this issue, two auxiliary textures are defined, which we call `auxiliaryCanvas(2)`. In `auxiliaryCanvas`, all pixels located within the crescent $\mathcal{C}(t_{d^*}) \setminus \mathcal{C}(t_{d^*-1})$ are assigned an *rgba*-value of $(1, 1, 1, 1)$, while all other *rgba*-vectors remain zero. In `auxiliaryCanvas2`, all pixels in $\mathcal{C}(t_{d^*-1})$ are assigned an *rgba*-vector of $(1, 1, 1, 1)$. If a pixel p in the first part of the stroke has a neighbor p_i without preimage coordinates in `preimageCanvas`, it is tested whether p_i has a value of 1 in its vector on `auxiliaryCanvas`. If this is the case, the preimage coordinate for p_i is stored in `preimageCanvas2` because it belongs to the neighborhood of p on the second part of a self-intersecting stroke. This preimage coordinate can be used in the usual way for the pixel averaging procedure of the Algorithm 6 for pixel p . Likewise, a pixel that has no preimage coordinates in `preimageCanvas2`, but has a non-vanishing *rgba*-vector in `auxiliaryCanvas2`, has preimage coordinates stored in `preimageCanvas`.

Further adjustments are necessary in handling the boundaries in the pixel averaging procedure from Section 5.2.2. The Schwarz reflection principle is still used for the first and last circles $\mathcal{C}(t_0)$ and $\mathcal{C}(t_N)$, which are now stored on two different textures. At the seam between the two parts of the stroke, the adapted generalization of the Schwarz reflection principle has to be applied to pixels in the boundary area, i.e., for a neighboring pixel p_i outside the stroke. If a pixel is located within either circle $\mathcal{C}(t_{d^*})$ or $\mathcal{C}(t_{d^*-1})$ and any of its neighbors p_i has no preimage coordinate that is stored in `preimageCanvas` for $d = d^* - 1$

or in `preimageCanvas2` for $d = d^*$, then it is first tested if p_i has a non-zero *rgba*-vector on `auxiliaryCanvas2` for $d = d^* - 1$ or on `auxiliaryCanvas` for $d = d^*$. If this is the case, the preimage coordinates for p_i stored on the respective other texture `preimageCanvas(2)` are used for pixel averaging. If not, the opposite neighboring pixel p'_i (see Figure 5.6) is also examined for the same properties, i.e., whether it lies within the stroke part to which pixel p also belongs. The usual procedure for pixels in the boundary of the stroke is applied if this is true. Otherwise, we check the respective other texture `auxiliaryCanvas(2)` to see if the opposite neighbor p'_i has a preimage there and potentially use it for the boundary handling procedure.

Two self-intersecting ornamental strokes generated from our algorithm are shown in Figure 5.16.

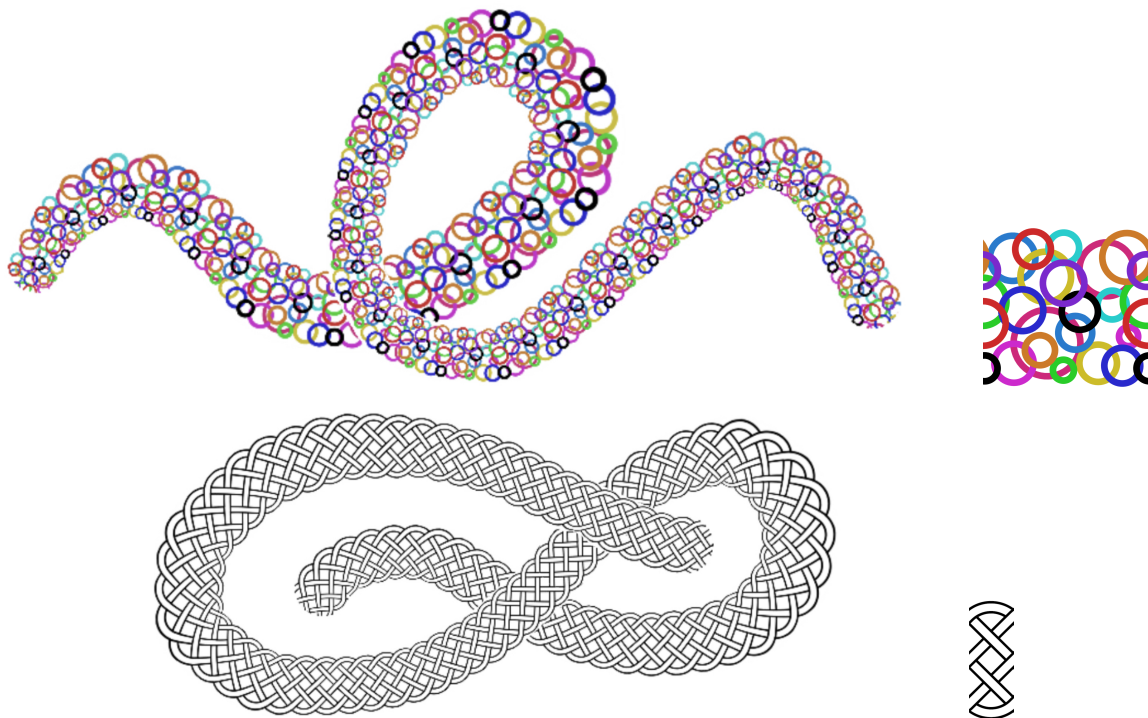


Figure 5.16: Strokes with self-intersection.

Also, the pixel-based local conformality test and the calculation of the stopping criterion need to be modified for the split stroke. A second texture `conformalTestCanvas2` is created that stores the deviation values for all pixels within the second part of the stroke. Since both methods rely on the neighbors of pixels, the same rules apply as for pixel averaging. For neighboring pixels p_i without preimage coordinates in one of the textures `preimageCanvas(2)`, the position on the respective other texture is checked. If the *rgba*-vector at p_i is non-zero there, the preimage coordinates are deduced from this other texture. Several strokes with displayed colors from the conformal test textures `conformalTestCanvas(2)` are shown in Figure 5.17 together with the global average deviation from local conformality according to our constraints.

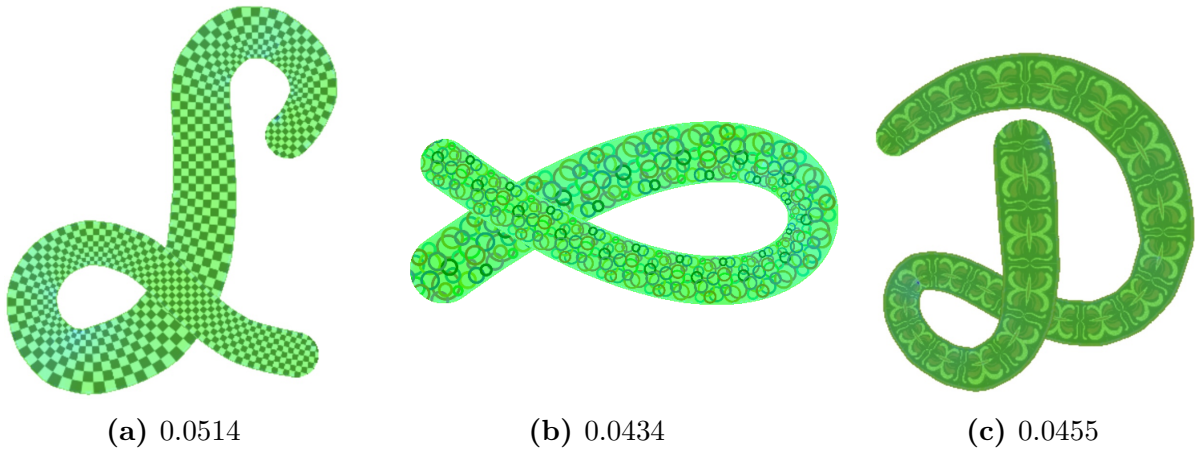


Figure 5.17: Self-intersecting strokes with visual and numeric test for local conformality.

5.4.2 Extension ideas for singular boundary points

Strokes also deviate from the univalent case if their boundary has points of regression. We have examined the continuous structure behind strokes with singular boundary points in Section 2.6. Singular points of type A_2 are known to be cusps by Proposition 2.6.8. A stroke can be considered as the projection of a swallowtail surface for both constant and non-constant radius functions in regions around points of γ with extreme curvature κ . We have already examined this projection in Section 4.3.2 and have presented methods for determining the positions of the two cusps S_1 and S_2 , as well as of the special points S_e and S_i , which are vertices of the two- and threefold regions within the stroke (see Figure 5.18).

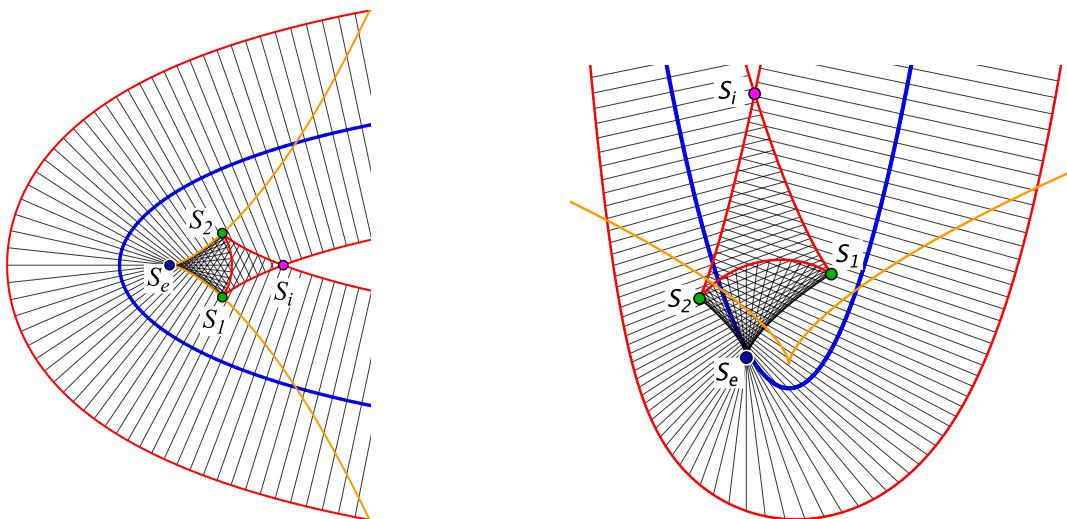


Figure 5.18: Benchmark Examples 2.1.9 and 2.1.10 with cusps and special points.

It was left open in Section 4.3.2 how to exactly compute the self-intersection point S_i of the envelope curve with cusp singularities as we want to follow the now known structure of the algorithm when doing so. For this, let w be the envelope curve containing the two cusp singularities S_1 and S_2 . The curve w changes direction when it meets S_1 and subsequently

only passes a region already covered by the stroke until it meets S_i . We compute S_i using the property that it is the point where the envelope curve intersects itself and leaves the already covered region of the drawing surface. As outlined in Section 5.1.2, each pixel inside the stroke contains a non-zero alpha-channel in its corresponding *rgba*-vector on the texture `centerCanvas`. Therefore, point S_i is located at $w(t_i)$ for some $t_i \in I$, where $w(t_i)$ meets a pixel p on the drawing surface that has a non-zero alpha-channel on `centerCanvas`, i.e., $(r, g, b, a)_p = (\bullet, \bullet, \bullet, 1)$. Additionally, it has to hold that, for some small constant $0 < \mu \ll 1$, the point $w(t_i + \mu)$ corresponds to a pixel in `centerCanvas` that has a zero alpha-channel since the envelope curve leaves the already covered stroke region just after passing S_i .

We include the search for S_i , as we did for the cusps S_1, S_2 and the point S_e , in the part of the algorithm that defines equidistant curve points on the interpolated piecewise spline curve γ (see Section 4.1.3). Hence, for parameters t_δ with distance $0 < \delta \ll 1$ from each other (the same δ as in Algorithm 1), we compute the boundary point $w(t_\delta)$. For each of these boundary points, we test if the alpha-channel of the *rgba*-vector in `centerCanvas` at $w(t_\delta)$ is equal to zero. In this case, the *previous* boundary point $w(t_\delta - \delta)$ is the point on w which corresponds to the point S_i .

If S_i is located exactly on the border between two spline pieces defining w , it would be necessary to use the previous spline piece to define $S_i = w(t_\delta - \delta)$. But all calculations throughout the algorithm are performed locally, meaning that only the current spline piece for w is available in each iteration. However, because we have chosen δ to be smaller than the width of a pixel, the positions of the two boundary points $w(t_\delta)$ and $w(t_\delta - \delta)$ cannot be distinguished on the texture underlying the stroke. Furthermore, it is possible that $w(t_\delta)$ is closer to the computational position of S_i than $w(t_\delta - \delta)$, since both points are only approximations of S_i . Hence, we use $w(t_\delta)$ instead of $w(t_\delta - \delta)$.

To save computing time, we search for the point S_i only if S_e has been detected. Furthermore, the subsequently determined position of S_2 is used as an indicator on which envelope curve the intersection point S_i lies.

Having identified the special points S_1, S_2, S_e , and S_i , we propose an approach to extend our algorithm from the previous sections of this chapter to also being able to handle strokes with singularities at the boundary. The concept is based on creating new textures for each layer of the two- and threefold covered stroke parts. For this, the stroke is split into several regions each of which is then stored on a separate texture. We propose to split the stroke into three different regions R_1, R_2 and R_3 , assuming that the stroke has no self-intersections or other singular boundary points except for the two cusps S_1 and S_2 . If the stroke contains additional cusps or self-intersections, the respective method has to be used again for the corresponding non-univalent part of the stroke. Figure 5.19 illustrates our proposed approach for Benchmark Example 2.1.9. The four pictures show the relevant details, while the beginning and end of the stroke are omitted. The approach equally applies to strokes with a non-constant radius function r as in Benchmark Example 2.1.10.

Without loss of generality, let the singular boundary points S_1 and S_2 be located on w_+ . The first region R_1 contains the first layer of the stroke, bounded on one side by the envelope curve $w_+(t)$ for $t \in [0, t_{S_1}]$, where $t_{S_1} \in I = [0, T]$ such that $w_+(t_{S_1}) = S_1$; S_1 is the first cusp on w_+ . On the other side, the region R_1 is bounded by the envelope $w_-(t)$

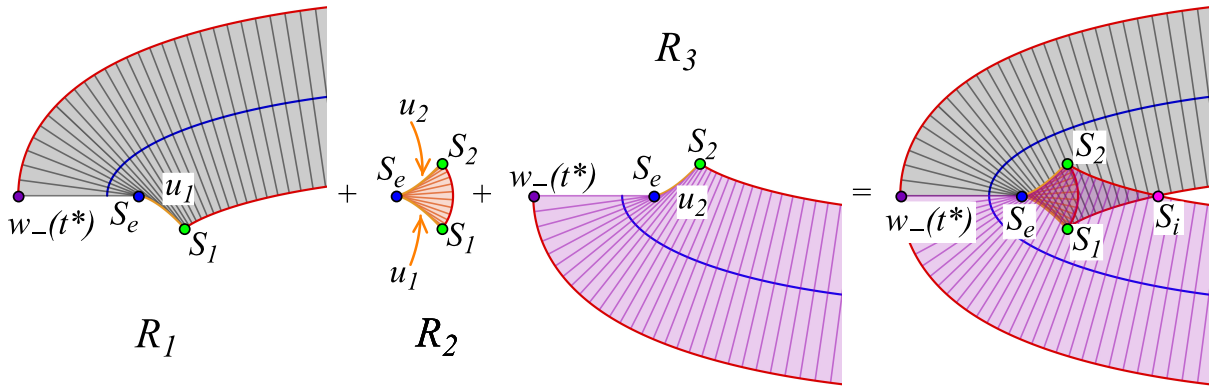


Figure 5.19: Splitting a multiply folded stroke into three disjoint regions.

for $t \in [0, t^*]$, where $t^* \in I$ such that γ has extreme curvature at t^* , i.e., $\kappa'(t^*) = 0$. The boundary of the region R_1 between $w_+(t_{S_1})$ and $w_-(t^*)$ consists of two segments: a line segment connecting S_e and $w_-(t^*)$, and a curve segment u_1 , which bounds the threefold covered region between S_1 and S_e . For a stroke of constant radius r_c , the curve u_1 that connects S_1 and S_e is the evolute ν of γ for the parameter $t \in (t_{S_1}, t^*)$ (see Lemma 4.3.1).

The second region R_2 represents one layer of the threefold covered region of the stroke. It is bounded by u_1 , the envelope curve w_+ between the two cusps S_1 and S_2 , i.e., $w_+(t)$ where $t \in (t_{S_1}, t_{S_2})$ for $t_{S_2} \in I$ with $w_+(t_{S_2}) = S_2$, and the curve segment u_2 connecting S_2 and S_e . If the stroke has a constant radius r_c , the curve u_2 is the segment of the evolute $\nu(t)$ of γ for the parameter $t \in (t^*, t_{S_2})$ (see Lemma 4.3.1).

The third region R_3 is bounded between S_2 and $w_-(t^*)$ by u_2 and a line segment connecting S_e and $w_-(t^*)$. It continues along the path of the stroke bounded by $w_-(t_-)$ for $t_- > t^*$ and $w_+(t_+)$ for $t_+ > t_{S_2}$.

Each of the three regions R_1 , R_2 and R_3 is stored using separate textures so that they can be superimposed on the drawing surface to display the final result. Furthermore, it is necessary to replicate all textures used to store the data required by the algorithm for each region.

It is part of future work to integrate this approach into the existing algorithm and to extend it for the cases not yet covered. There are some challenges that need to be overcome before the implementation.

As stated in Section 4.3.2, it is still open how to identify the curves u_1 and u_2 that connect S_e to S_1 and S_2 for a stroke with non-constant radius function r .

Another challenge is that a user-drawn curve is usually drawn wobbly and with irregular pressure changes. Thus, unlike Benchmark Examples 2.1.9 and 2.1.10, the equation of Proposition 2.5.2, which we use to detect singular boundary points, can be satisfied for more than the obvious spots at narrow turns, i.e., around points of extreme curvature of the stroke. This is visualized in Figure 5.20, which shows the family of circles behind some hand-drawn strokes. The green points indicate registered cusps, while the blue points represent determined points S_e , and the pink points are intersections S_i of the envelope curves with themselves. The stroke on the left shows that the test for singular boundary points found more green cusps than the two cusps related to the blue point S_e and the pink point S_i . Furthermore, defining universal small constants, which serve as

measures for detecting singular boundary points and S_e or S_i , continues to be a challenging task. We notice that the search for points S_e and S_i in the right stroke in Figure 5.20 did not find all of them.

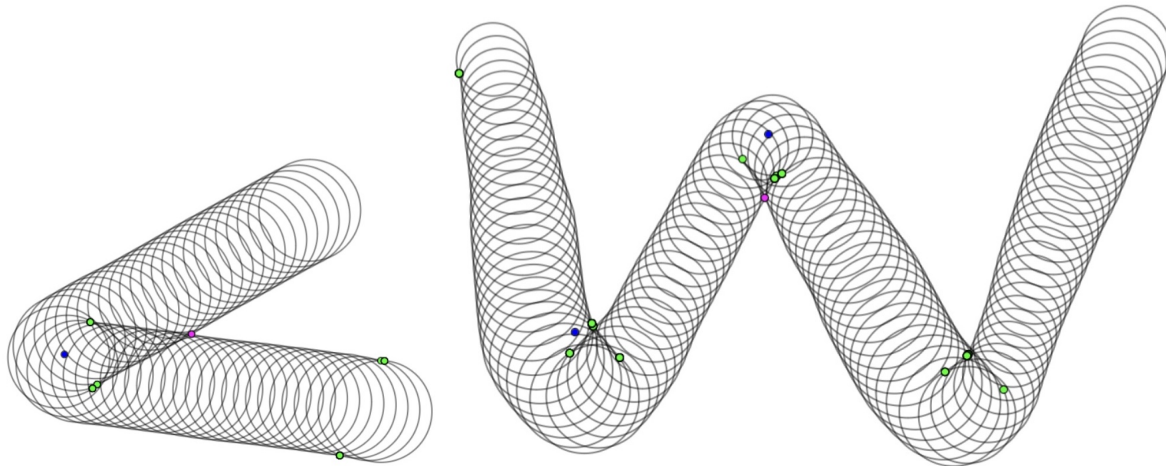


Figure 5.20: Erroneous detection of cusps and the related special points.

Another challenge in implementing the division of the stroke into different layers is caused by the fact that our algorithm adds circular crescents to the stroke region for each new circle $\mathcal{C}(t_d)$. Therefore, the algorithm may have already saved preimage coordinates for pixels on a texture corresponding to a layer that will no longer belong to that layer after the stroke is split. This preimage data needs to be correctly transferred to the next layer. However, computing the correct fold along the curves u_1 and u_2 is a major challenge. The seam between R_1 and R_3 along the line connecting $w_-(t^*)$ and S_e is not critical, since there is no fold and the preimage coordinates can be transferred without further calculations.

The correct folding functions are also needed to identify the neighbors of pixels near u_1 and u_2 , which are required for pixel averaging. We expect the resolution of the close neighborhood around S_e , where all three regions meet, to be particularly challenging.

6 | Application: Ornamental Design

In the previous chapter, we have presented our algorithm which computes a real-time approximation of a conformal map from an ornamental strip to a user-drawn stroke. All computations are executed on the GPU, and both the stroke and strip regions are stored as textures. The algorithm is based on pixel averaging, where the preimage coordinates for each pixel within the stroke are computed by iteratively averaging over the preimage coordinates of its neighbors. This method makes the coordinate system, which was conformally transformed from the strip to the stroke, converge to the solution. Reverse pixel lookup is applied to transfer the artistic content of the strip texture to the stroke texture.

Instead of referring to the artistic content of an ornamental tile, we could also directly use the transformed coordinate system stored in the texture `preimageCanvas` to decorate the stroke. For this, we replace the line 7 in Algorithm 5, which reads a color from `tileCanvas` at the preimage coordinates `coord = (x, y)` of pixel p , with rules based on these preimage coordinates. An elementary example is to color a pixel of the stroke in a transparent red, i.e., to set $(r, g, b, a)_p = (1, 0, 0, 0.5)$, if $y \bmod \frac{h_T}{8} > \frac{h_T}{16}$. All other pixels remain fully transparent. The result is a striped stroke as shown in Figure 6.1.

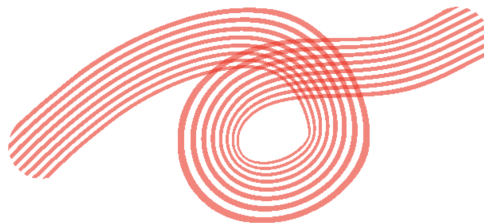


Figure 6.1: Simple example of creating visual output based on coordinates.

For a striped design as in Figure 6.1, it is necessary to specify a reference height h_T even though no preimage tile is used. Similarly for designs with a variation along the drawing direction of the stroke, it is necessary to specify a reference width w_T in which the design might repeat. This is the case in Figure 6.2 on the left, where the design repeats every four stars and circles. Once a reference height h_T and width w_T have been specified, any design can be developed based on rules for the coordinate system stored in the stroke's pixel information. This works well as long as the modulo calculations of the preimage coordinates are respected when writing the rules. However, there is also a drawback that we will illustrate below with reference to the design consisting of three stripes, circles, and stars, as in the stroke in Figure 6.2 on the left.

Since the strokes are modeled for pressure-sensitive digital pens, the defining circles have variable radii that are determined by the pressure applied, or by the user's choice

for non-pressure-sensitive drawings (see Section 5.1). Thus, also the design has to be suitable for both narrow and wide strokes. Theoretically, this is not a problem since the colors are assigned correctly no matter how narrow the stroke is. However, for small radii, i.e., narrow stroke parts, the design of stars and circles is hard to recognize, as in Figure 6.2 on the right, where the design looks pixelated and the individual shapes are badly recognizable.

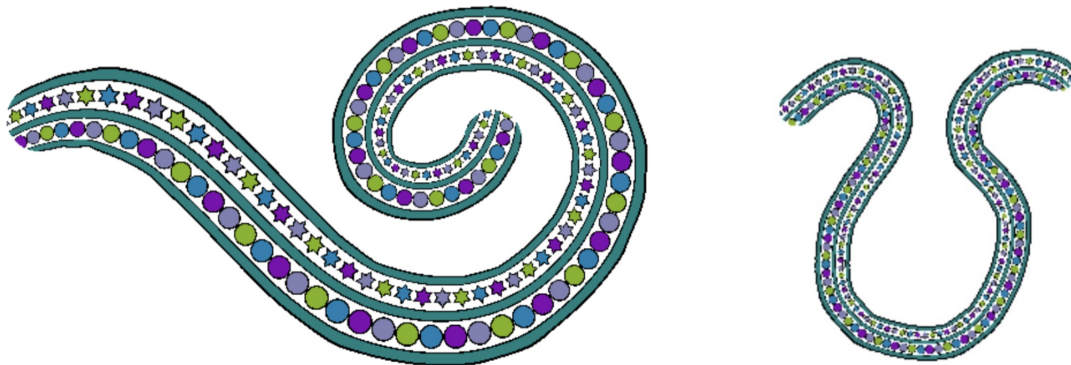


Figure 6.2: Drawback of using coordinate based rules: no MIP mapping.

This effect also occurs when using the algorithm presented in Chapter 5, but there it can be reduced when doing pixel lookup from a tile texture, since the GPU computations allow the use of *MIP mapping*. The MIP mapping technique computes downsampled versions of a texture, which are then displayed according to the pixel resolution of the regions in which these textures are displayed. Using a downsampled version in an area of low pixel resolution reduces the aliasing effect that makes the design look pixelated and partially incomplete, as if pixels have been cut out. Similar to the `interpolate` modifier (see Section 5.1.3), the modifier `mipmap → true` is added to the CindyJS code in line 7 of Algorithm 5 to activate MIP mapping.

However, when we use the coordinate information from `preimageCanvas` directly to generate a design for the stroke, we are not using the reverse pixel lookup from a tile texture and therefore cannot use the `mipmap` modifier. One way to address the inability to use MIP mapping is to implement a rule that omits design details when the radius of the circle containing the pixel falls below a specific value. Possible variations of the design consisting of stars and circles are shown in Figure 6.3 on the left: for wide strokes, the original design is used. For more narrow stroke parts, the stars are replaced by circles which contain less detail since they don't have star tips. In the case of very narrow strokes, all circles are replaced by colored squares, and all black contours are omitted.

Whether a stroke is classified as wide or narrow depends on the observer, or on the limits imposed on the radii of the stroke-defining circles by the creator of the design rules. If the radius value of the circle to which a pixel belongs exceeds or falls below a certain threshold, the variant with more or fewer details is used to determine the pixel's color.

This radius-based design variation works perfectly for strokes of constant width. However, if the width of a stroke changes, the design elements are truncated since the stroke gradually grows by circular crescents of varying radii. This is illustrated in Figure 6.3 on the right, where three highlighted regions show design elements that are truncated due to the change of the radii between two neighboring circular crescents.

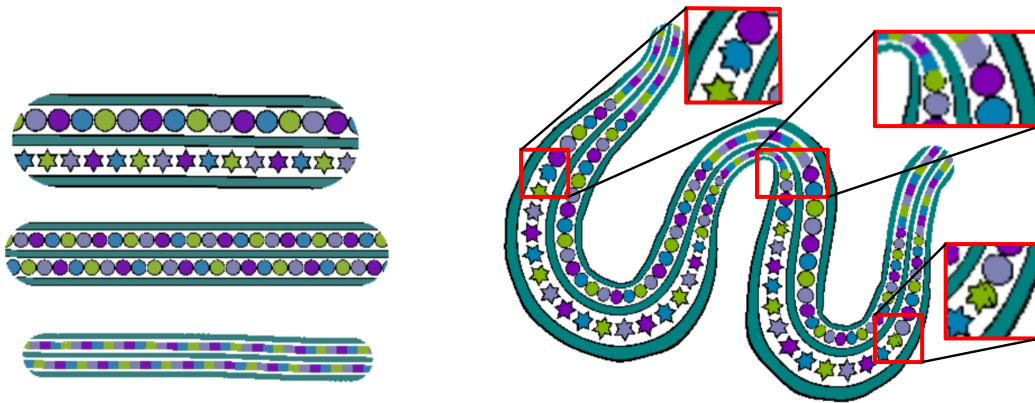


Figure 6.3: Pressure-sensitive design and broken stars.

To overcome this problem, we will present a variation of our algorithm from Chapter 5 that computes which design unit of height h_T and width w_T a pixel in the stroke belongs to (see Section 6.1). With this approach, we will use different levels of detail in the units, depending on how wide or narrow the stroke is in those units (see Section 6.2.1). Furthermore, we will apply design transitions between wider and narrower parts of the stroke and we will use a start and an end tile (see Section 6.2.2). Moreover, Sections 6.2.2 and 6.2.3 show ways to create more variation in the design, even for strokes of constant width.

The complete code for the HTML applet, entitled *DesignStrokes_LP.html* and created in CindyJS, is available on [Pol23b].

6.1 Tile recognition in the algorithm

Let h_T be the height and w_T the width of a design unit that is supposed to decorate the stroke. In contrast to the algorithm presented in Chapter 5, we no longer store the preimage coordinates as $\text{preimpt} = (x \bmod m \cdot h_T, y)$ for each new circular crescent in the following. Instead, we store the x -coordinate of the preimage of the pixel p modulo w_T and, additionally, the information in which design unit T_k the pixel's preimage is located for

$$k = \left\lfloor \frac{x}{w_T} \right\rfloor + 1,$$

where $k \geq 1$ for computational reasons. When using the coordinates for calculations, both pieces of information are combined and the calculations are performed on these coordinates. This means that the calculations are no longer performed on a shortened preimage strip S_m of width $m \cdot h_T$ (see Section 5.1.1), but on the complete tiled strip S_n consisting of n design units T_k for $k \in \{1, \dots, n\}$.

The preimage coordinates preimpt of a pixel p in the stroke are still calculated by the piecewise defined function τ_d as in the Equations (5.3). But the subsequent adaptations due to calculations modulo $m \cdot h_T$ are neglected, and $\tau_d(\gamma(t_d)) = (x_d, \frac{h_T}{2})$ with x_d as in Equation (5.2) is also not taken modulo $m \cdot h_T$. The computed preimage coordinates are taken modulo w_T and then stored as usual in the texture `preimageCanvas` with the

function `splitData`. Hence, in Algorithm 4, line 9 is modified to

$$\text{rgba} = \text{splitData} ((\text{mod}(\text{preimpt_1}, w_T), \text{preimpt_2}, 0, 1), \text{scvel}, \text{base}); \quad (6.1)$$

Additionally, a new texture is created, which we call `tileCanvas`. In this texture, for every pixel in the stroke, we store the number k of the design unit T_k the pixel belongs to. The value of $k - 1$ corresponds to the multiple of w_T that was cut off `preimpt` due to calculations modulo w_T . This means that the *rgba*-value for a pixel p in the stroke is set to

$$(r, g, b, a)_p = \left(1 + \left\lfloor \frac{\text{preimpt_1}}{w_T} \right\rfloor, 0, 0, 1 \right) = (k, 0, 0, 1).$$

Depending on the use case, a variation of `splitData` can be defined for the texture `tileCanvas`, which splits the integer k into two different channels of the *rgba*-vector, making it possible to store more than 256 different numbers k . For this, the same adaptations are necessary as for `equiCanvas`, which stores the amount d of the circles $\mathcal{C}(t_d)$ (see Section 5.1.2 below Algorithm 4). For now, we assume that 256 design units are sufficient and will not make any further adaptations.

In addition to the changed storage of the preimage coordinates modulo w_T in `preimageCanvas`, the x -coordinate of the preimage of a circle center $\tau_d(\gamma(t_d))$ is stored modulo w_T in the texture `centerCanvas`, while its y -coordinate remains $\frac{h_T}{2}$ without exception. However, additional information is needed to recover the exact coordinates of the preimage circle centers. In general, adding the multiple of w_T stored in `tileCanvas` to the x -coordinate in `centerCanvas` is incorrect. This is because texture `tileCanvas` contains the number k of the design unit T_k in which pixel p lies, but its assigned preimage circle center $\tau_d(\gamma(t_d))$ probably lies in a different design unit $T_{\tilde{k}}$ with $k \neq \tilde{k}$. Hence, we use the blue-channel of the *rgba*-vector in `centerCanvas` to store the correct number $\tilde{k} = \left\lfloor \frac{\tau_d(\gamma(t_d))}{w_T} \right\rfloor$ for the preimage circle center. Let $\tau_d(\gamma(t_d)) = (p_x, p_y) = (p_x, \frac{h_T}{2})$, then for every pixel p of `centerCanvas` inside the stroke, the *rgba*-vector is set to

$$(r, g, b, a)_p = \left(\text{mod}(p_x, w_T), p_y, \left\lfloor \frac{p_x}{w_T} \right\rfloor, 1 \right).$$

Due to the new way of storing the preimage coordinates and the preimage circle centers modulo w_T in the respective textures, the pixel averaging algorithm of Section 5.2 has to be adapted slightly. The main algorithmic concept remains unchanged. The arithmetic mean of the preimage coordinates of the neighbors p_i is calculated for each pixel p and then stored as the new preimage in the *rgba*-vector of p (see Section 5.2). Also, the boundary handling with the Schwarz reflection principle for the first and last circles as well as the adapted generalized version for reflection at the analytic boundary curves remain the same (see Section 5.2.2). For correct calculations, the preimage coordinates `preimcoord` of a neighboring pixel p_i have to be restored from the information in the textures `preimageCanvas` and `tileCanvas` as follows:

```
preimpt = restoreData ( imagergba ( (0,0), (1024,0), "preimageCanvas", p_i ), scvel, base);
tilenum = imagergba( (0,0), (1024,0), "tileCanvas", p_i ) - 1;
preimcoord = (tilenum · w_T + preimpt_1, preimpt_2);
```

To compute the value of `tilenum`, we subtract 1 from the stored value $k \geq 1$. When circular reflection is applied as part of the boundary handling using the preimage circle center `preimcenter`, its coordinates are read from the texture `centerCanvas` by

```
center = restoreData ( imagergba( (0,0), (1024,0), "centerCanvas", p_i ), scvel, base);
tilenum = center_3;
preimcenter = (tilenum * w_T + center_1, center_2);
```

When storing the updated coordinates `preimnew` of a pixel p to the texture `preimageCanvas` after averaging, it is generally not possible to store the vector as in Equation (6.1) because calculating `preimnew_1` modulo w_T could cause a loss of information. This loss of information occurs when pixel averaging shifts a newly calculated preimage point to another tile or design unit if it was already very close to it before. All new preimage coordinates in `preimageCanvas` are computed within a `colorplot` function environment, which performs all calculations on the GPU in parallel for all pixels in `preimageCanvas`. Hence, it is not possible to simultaneously update the entries of `tileCanvas`. This means that we have to store the difference between the unit number k from `tileCanvas` and the actual value together with the new coordinates `preimnew` in `preimageCanvas` to access the correct coordinates for further calculations later on:

```
tilenum_p = imagergba ( (0,0), (1024,0), "tileCanvas", p )_1 - 1;
rgba = splitData ( ( preimnew_1 - tilenum_p, preimpt_2, 0, 1), scvel, base);
```

Finally, it is necessary to calculate all preimage coordinates used in the test for conformality presented in Section 5.3.1 using the stored number k in `tileCanvas` and the coordinates from `preimageCanvas`.

In the next section, we will apply the new approach of storing preimage information modulo w_T and the number of the design unit T_k to which a pixel belongs. The new procedure for storing preimage coordinates allows to modify the reverse pixel lookup from Section 5.1.3.

6.2 Use cases of variable tiles

To solve the issue that the design for strokes of different widths cuts off design elements when the design is selected depending on the local width of the stroke (see Figure 6.3), we have adapted the algorithm to store the number k of the design unit T_k to which a pixel of the stroke belongs. However, exclusively storing the preimage coordinates differently than in Section 6.1 for all pixels of the stroke does not solve this problem. In addition, we need to apply the same design rules to all pixels belonging to the same design unit. For this, another texture is created, which we call `ruleCanvas`. This texture operates as a 1024×512 matrix, where each entry is a 4-dimensional *rgba*-vector. Depending on the particular use case, rule information for different design units can be stored using columns, rows, or even smaller units of the matrix. For simplicity, we assume that there are no more than 1024 design units, which permits using only the columns of the texture `ruleCanvas` to store information. Whenever a new design unit T_k is reached in the stroke, the k^{th} column of `ruleCanvas` is updated to store the new information in a pixel $p = (x, y)$ if its x -coordinate is in the interval $(k - 0.5, k + 0.5)$. This is the reason why we started counting design units with $k = 1$.

In the part of the algorithm where we previously used reverse pixel lookup to read the color information for each pixel in the stroke from the shortened strip texture S_m , we now change our approach to instead read the rule for the tile T_k using the pixel's corresponding tile number k from the texture `tileCanvas`. The rule is stored in the texture `ruleCanvas` at position (k, y) for $y \in [0, 512]$. The color of each pixel is defined according to this rule. What the rules stored in `ruleCanvas` should look like depends on the use case, as will be illustrated in the following sections.

6.2.1 Floral design: radius dependent textures

The first example is a floral design based on the four different image tiles *floral1* to *floral4* from Figure 6.4.

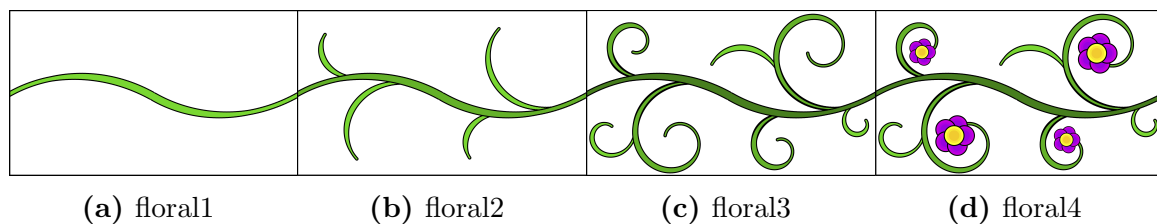


Figure 6.4: Four different textures of floral design.

Depending on the stroke width that corresponds to one design unit T_k , we select an image tile and access the color from this tile using reverse pixel lookup. The radius $r(t_d)$ of the defining circles $\mathcal{C}(t_d)$ equals half of the stroke width. Since the radii are already stored in `radiusCanvas`, we use this information further on. The range of radius values is $[10, 40]$ (see Section 5.1). We establish the rule to use *floral1* for the reverse pixel lookup in a design unit if the radius is between 10 and 15, *floral2* for a radius between 15 and 20, *floral3* for a radius between 20 and 30, and *floral4* for a radius greater than 30.

The question is, which pixel within the design unit defines which radius is the decisive one and, thus, which floral design the rest of the unit has.

To define this, we specify when and how to update `ruleCanvas`. We determine when a new rule for a new design unit T_k needs to be specified depending on the progress of the preimage circle around $\tau_d(\gamma(t_d))$ on the horizontal line $y = \frac{h_T}{2}$ in the preimage strip S_n as given in Equation (5.2). When the stroke starts, the first circle $\mathcal{C}(t_0)$ covers $\frac{h_T}{w_T}$ design units because the preimage circles all have a diameter of h_T and the design unit has a width of w_T . For each started design unit, i.e., for T_k with $k \in \left\{1, \dots, \left\lceil \frac{h_T}{w_T} \right\rceil\right\}$, the corresponding column k in the texture `ruleCanvas` is assigned the *rgba*-vector $(\mathbf{r}, 0, 0, 1)$, with $\mathbf{r} \in \{1, 2, 3, 4\}$ depending on the radius $r(t_0)$ of the first circle $\mathcal{C}(t_0)$. The variable \mathbf{r} represents the floral image tiles numbered 1 through 4. If $r(t_0) > 30$, \mathbf{r} is set to 4; if $30 > r(t_0) > 20$, \mathbf{r} is set to 3, and so on. When a new circular crescent of $\mathcal{C}(t_d)$ is added to the stroke after the first circle, pixels within this newly added crescent are tested if they are contained in a new design unit T_{k+1} . In this case, a new rule is needed for those pixels. To compute the number k of the design unit T_k a pixel is contained in, the amount $x_d - x_{d-1} = \frac{h_T \cdot |\gamma(t_d) - \gamma(t_{d-1})|}{2 \cdot r(t_d)}$ with $x_0 = \frac{h_T}{2}$ (see Equation (5.2)) is added to $\frac{h_T}{w_T}$ every time a new circle is appended to the stroke. Hence, the number of started design units is given by $\left\lceil \frac{h_T}{w_T} + \left(x_d - \frac{h_T}{2}\right) \right\rceil$. If this number increases by one, a new design unit T_{k+1} is reached and a new rule depending on the radius $r(t_d)$ of the current circle $\mathcal{C}(t_d)$ is stored to the $(k+1)^{\text{th}}$ column of `ruleCanvas`. This means that the circle $\mathcal{C}(t_d)$ that first reaches a new design unit sets the rule for the rest of the tile. For strokes that increase in width, this causes the image tiles intended for narrow parts of the stroke to appear in parts of the stroke that are wider than the image tile was intended for. For parts of decreasing stroke width, it is the other way around.

In the reverse pixel lookup, the number of the design unit T_k to which each pixel p in the stroke belongs is accessed by the texture `tileCanvas`. At the corresponding column k of texture `ruleCanvas`, the number $\mathbf{r} \in \{1, 2, 3, 4\}$ determines the preimage texture from which the pixel's color has to be read: 1 for *floral1*, 2 for *floral2*, 3 for *floral3*, and 4 for *floral4*. Figures 6.5 and 6.6 show some strokes created with these design rules.



Figure 6.5: Stroke with floral design varying with the stroke's width.

There are three remarkable specialties in the strokes of Figure 6.6. First, both strokes have self-intersections, which are treated as usual with a new texture for the second part of the stroke displayed on top of the first part. A self-intersecting floral design like this brings up the idea of merging the designs at self-intersections of strokes. An approach for this is not part of this thesis, but the selection of future research ideas in Chapter 7 revisits this topic. Secondly, the stroke on the right in Figure 6.6 is nearly closing. It may be desirable to adjust the stroke slightly at both ends so that the stroke actually closes. At the same time, the design would also need to be adjusted. This also remains an open problem within this thesis and will be addressed again in Chapter 7. Thirdly, the design of the floral pattern is truncated at the end of the right stroke, resulting in two incomplete branches. To avoid this, it is possible to include rules for the design at the end of a stroke. Such rules will be presented in the following Section 6.2.2 for a braid design.



Figure 6.6: Stroke with floral design varying with the stroke's width.

Note 6.2.1 *The floral design of this section was created using tangential logarithmic spirals. We present the necessary theory behind tangentiality of logarithmic spirals in Appendix B, followed by a CPU-based algorithm for generating strokes of tangential logarithmic spirals.*

6.2.2 Braid design: transitions and variations

In this section, we study the braid design consisting of the image tiles in Figure 6.7.

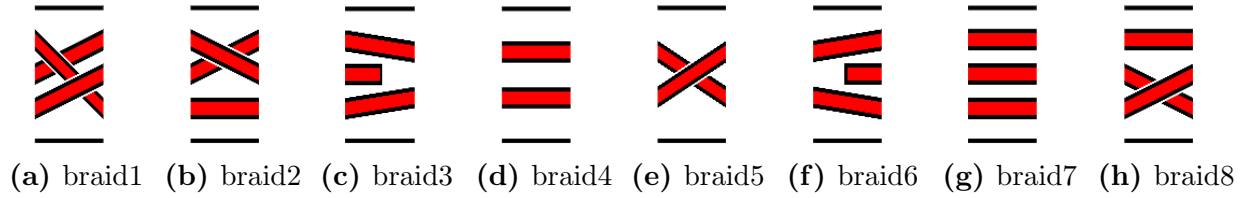


Figure 6.7: Textures of braid design.

There is a significant difference between the braid tiles and the floral tiles in the previous section: while the edges of the floral tiles match seamlessly, so that all floral tiles can be combined, there are braid tiles that do not match, such as *braid1* and *braid4*. The braid tiles 1, 2, 7, and 8 are designated for use in stroke parts with large width, while braid tiles 4 and 5 are intended for narrow stroke parts. For a seamless transition between these parts, we need to use the image tiles *braid3* or *braid6*, depending on whether the width of the stroke increases or decreases.

To detect a transition between a narrow and wide part of the stroke, a new step is added to the algorithm, which checks the content of column $k - 1$ in `ruleCanvas` whenever column k of `ruleCanvas` is updated because a new design unit T_k has been reached by the stroke. If the stroke is narrow according to the preset design rules when it reaches T_k , one of the image tiles 4 and 5 has to be used as a preimage for all pixels in T_k . But if column $k - 1$ in `ruleCanvas` indicates that the last design unit T_{k-1} was assigned one of the image tiles 1, 2, 7 or 8 for wide parts of the stroke, the design unit T_k is assigned *braid3* as preimage instead of *braid4* or *braid5*. An analogous test assigns *braid6* to the first design unit of a wide part of the stroke if the previous design unit belongs to the narrow part of the stroke.

For narrow stroke parts, both *braid4* and *braid5* can be used as design units, while braid tiles 1, 2, 7, and 8 are alternatives for design units within wide stroke parts. The presented braid tiles do not contain alternatives for the tiles 3 and 6, but additional tiles could easily be added. Thus, we have four distinct groups of image tiles: group 1 contains tiles for wide stroke parts, group 2 contains tiles for the transition from wide to narrow stroke parts, group 3 contains tiles for narrow stroke parts, and group 4 contains tiles for the transition from a narrow to a wide stroke part. One possibility would be to select a tile from the respective group for a new design unit T_k by cyclically rotating between the alternatives. Another option is to randomly select one image tile per group. Figure 6.8 shows an example where the braid tiles were randomly selected from their respective groups.

The groups are also helpful to give a clearer structure to the reverse pixel lookup. In Section 6.2.1, it was defined that column k of `ruleCanvas` stores the number `r` of the image tile assigned to the design unit T_k . If there are a large amount of image tiles, for example due to many alternatives, it is more convenient to store the group number of the image tile in the red-channel of all pixels in column k of `ruleCanvas`. Furthermore, the number corresponding to the tile of the group is stored in the green-channel. For

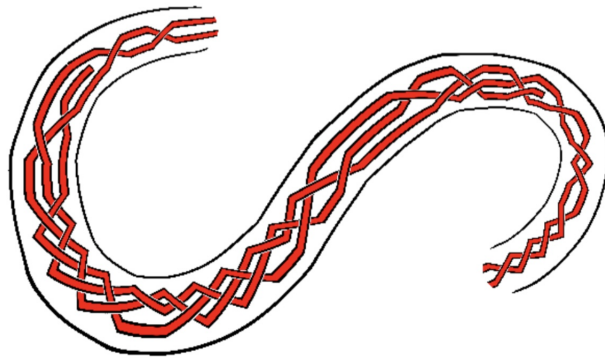


Figure 6.8: Stroke with variable braid design.

instance, if design unit T_k is assigned an image tile from group 1, then $\mathbf{r} = 1$ and the $rgba$ -vector in the k^{th} column of `ruleCanvas` is assigned the vector $(r, g, b, a)_k = (1, z, 0, 1)$ with $z \in \{1, 2, 7, 8\}$. See Algorithm 10 for the pseudocode of storing the number of randomly chosen tile alternatives and transition tiles in `ruleCanvas`. The transition from a wide to a narrow stroke occurs at a radius value threshold of 25.

Algorithm 10 Random alternatives in `ruleCanvas` along with transition tiles.

```

if a new design unit  $T_k$  was started in the stroke with circle  $\mathcal{C}(t_d)$  then
  forall pixels  $p = (x, y)$  in ruleCanvas with  $k - 0.5 < x \leq k + 0.5$  do
    if  $r(t_d) > 25$  then
      lastGroup  $\leftarrow$  red-channel of a pixel in column  $k - 1$  of ruleCanvas
      if  $k > 1$  and lastGroup == 3 then  $(r, g, b, a)_p \leftarrow (4, 0, 0, 1)$ 
      else randomNumber  $\leftarrow$  random number between 0 and 1
        depending on randomNumber choose an alternative  $z \in \{1, 2, 7, 8\}$ 
         $(r, g, b, a)_p \leftarrow (1, z, 0, 1)$ 
      end if
    else for  $r(t_d) \leq 25$ 
      lastGroup  $\leftarrow$  red-channel of a pixel in column  $k - 1$  of ruleCanvas
      if  $k > 1$  and lastGroup == 1 then  $(r, g, b, a)_p \leftarrow (2, 0, 0, 1)$ 
      else randomNumber  $\leftarrow$  random number between 0 and 1
        depending on randomNumber choose an alternative  $w \in \{4, 5\}$ 
         $(r, g, b, a)_p \leftarrow (3, w, 0, 1)$ 
      end if
    end if
  end forall
end if

```

The stroke in Figure 6.8 does not have a satisfactory design at its ends because the strands of the braid, as well as the black border at the top and bottom of the design, end without closing. A similar issue has already been mentioned in Section 6.2.1.

To resolve this, we include four additional image tiles for the beginning and end of a stroke. These braid tiles are shown in Figure 6.9.



Figure 6.9: Braid tiles for the beginning and end of a stroke.

The new image tiles for the beginning of the stroke are integrated into the existing rule system by automatically assigning one of the two alternatives as the first preimage tile to the design unit T_1 . Which one is assigned depends on the stroke width. If the first design unit belongs to group 1, the left start tile is used. If the first design unit belongs to group 3, the right start tile is used.

The two image tiles for the end of the stroke are not used while the user is drawing the stroke. Once the user stops drawing, the last complete design unit is determined. Assume the last completed design unit is T_{N-1} . Then, the reverse pixel lookup assigns the vector $(0, 0, 0, 0)$ to all pixels in T_N , leaving them fully transparent without any design. For all pixels in the design unit T_{N-1} , the color is read by the reverse pixel lookup from one of the end tiles depending on the stored group number in column $N - 1$ within `ruleCanvas`.

If both the start and end tiles are used, the result is a stroke with a design that closes at both ends of the stroke (see Figure 6.10). However, this process results in a sudden change of the design at the end of a just completed stroke. Additionally, the stroke is shortened by the last incomplete design unit.

An alternative could be to use the end tiles during the drawing process for the last complete or incomplete design unit. But this would not only lead to sudden design changes when the stroke is completed. Instead, when a new design unit is opened, the role of the “last” design unit would be transferred to another unit T_k . As a result, the sudden changes would be constantly present instead of once when the stroke is completed.

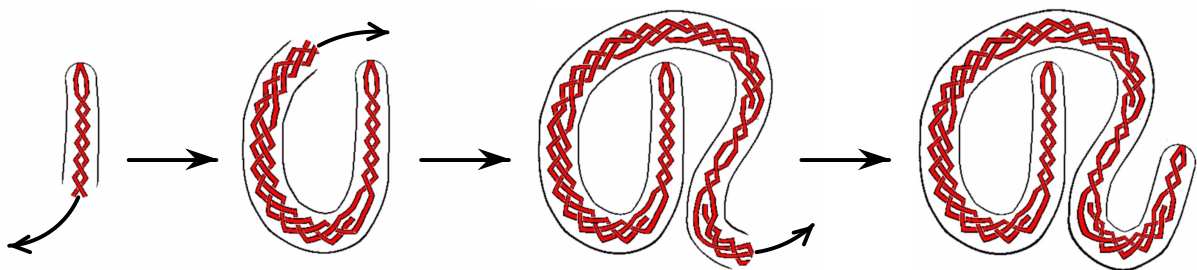


Figure 6.10: Drawing a stroke with special image tiles for the start and end of the stroke.

Note 6.2.2 Comparing Figures 6.8 and 6.10, it becomes apparent that Figure 6.10 contains fewer alternative image tiles in the wide parts of the stroke. This is because when using the four additional image tiles for the start and end of the stroke, the large number of different textures called in the reverse pixel lookup exceeds the maximum number of textures a `colorplot` function can handle in `CindyJS`. We leave the task of finding a way to allow a larger number of textures within a `colorplot` function to future research.

6.2.3 Further design concepts

We present two design concepts for strokes of constant radius function. The image tiles used for these designs are the tiles *circles1* through *circles4* shown in Figure 6.11. All four image tiles show tangential Apollonian circles with different iteration depths. The basic rules for a stroke design are to use *circles4* for strokes with a radius greater than 30 (group 1), to use *circles3* for a radius between 20 and 30 (group 2), *circles2* for a radius greater than 15 and less than 20 (group 3), and *circles1* for all radii smaller than 15 (group 4).

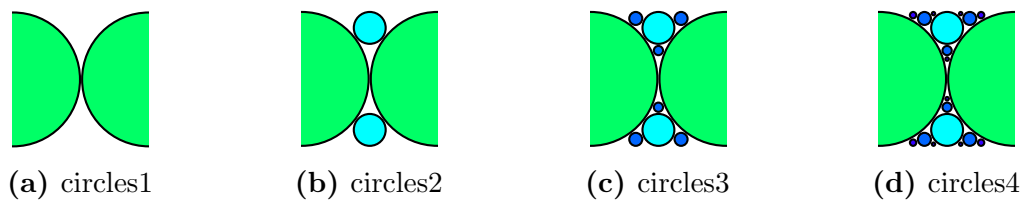


Figure 6.11: Four different textures with tangent Apollonian circles.

Our goal is to equip strokes of constant radius function with more variability in their design without creating alternative image tiles per group. For this, we allow the image tiles as alternatives for all groups with a higher number than the number of the group they originally belong to. In our example, the image tile *circles1* could be used for all groups, *circles2* for groups 2, 3, and 4, *circles3* for groups 3 and 4, and *circles4* for group 4 only. Selecting one of the possible alternatives per group can again be done randomly, as shown in Figure 6.12.

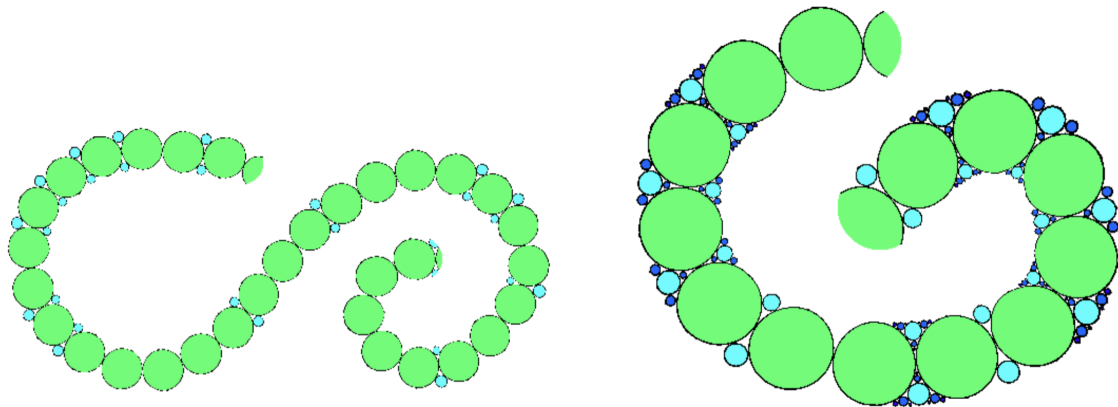


Figure 6.12: Strokes of different constant widths and variable design of tangent circles.

Another non-random option is to use the most detailed image tiles in the design units towards the middle of the stroke and to gradually reduce the level of detail in the image tiles used for the design units towards the ends of the stroke. Figure 6.13 shows an example.

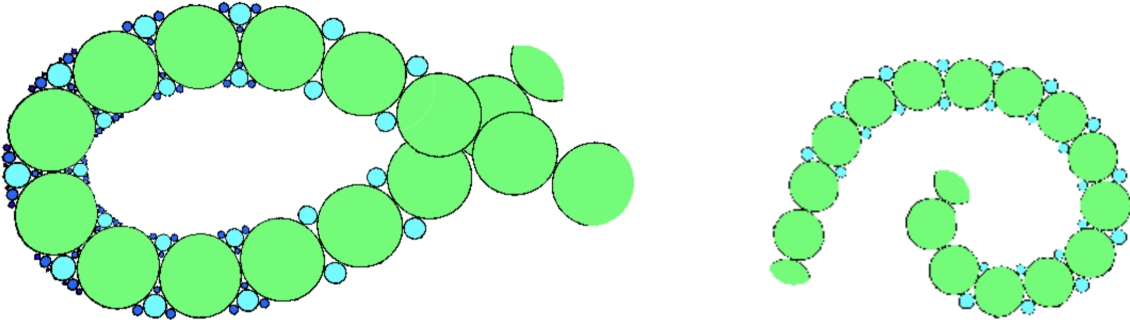


Figure 6.13: Strokes of constant width and design depending on their lengths.

As a conclusion, we present a design concept based on the idea of mapping text to a stroke, keeping the text in the middle of the stroke, and filling the parts of the stroke between the text and the ends of the stroke with additional ornamental design. If the length-ratio of the text and the ornament is h_T/w_T , the presented segmentation of the stroke into design units can be used in such a way that the design unit(s) in the center of the stroke are assigned the text tile as a preimage and all other units are assigned the ornamental tile. Doing this may cause the text to shift to one side of the stroke due to the design units not subdividing the stroke properly to keep the text in the center of the stroke.

Alternatively, we use only the method of storing the preimage coordinates from Section 6.1, i.e., we use the information from the texture `tileCanvas` and the coordinates from the texture `preimageCanvas` to access the exact preimage coordinates for all pixels in the stroke. With those exact preimage coordinates, it is possible to keep the text in the center of the stroke and fill the ends symmetrically with ornamental design. Figure 6.14 shows a stroke decorated with text together with an ornamental design of logarithmic spirals.

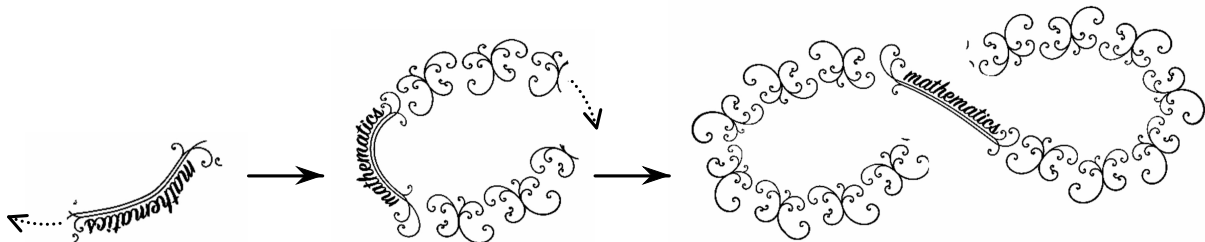


Figure 6.14: Stroke with text and ornamentation.

We assume that both the text and ornamental design are of a height of h_T . If not, they are scaled accordingly. Let the text have a width of w_T . To position the text in the middle of the stroke, we use the preimage coordinates of the last registered circle center $\tau_d(\gamma(t_d)) = (x_d, \frac{h_T}{2})$. The center of the stroke is where the pixels' preimage coordinates have an x -coordinate of $x_d/2$. Hence, for all pixels with preimage coordinates (x, y) where $x \in (\frac{x_d}{2} - \frac{w_T}{2}, \frac{x_d}{2} + \frac{w_T}{2})$ and $0 \leq y \leq h_T$, the text is displayed in that part of the stroke. For all remaining pixels, the ornamental design is used as preimage. It is not necessary that the width of the text is the same as the width of the ornamental tile.

7 | Future Research

Throughout the thesis, several interesting open research questions have been mentioned. For example, we consider it relevant to study all cases where singularities occur on the envelope curves for strokes with non-constant radius functions r . So far, we have only examined Benchmark Example 2.1.10, which contains a pair of cusps related to an extremum of curvature of the curve γ . However, Section 2.6.2 demonstrates that $A_{\geq 3}$ singularities and possibly related cusps may also appear at inflection points of γ .

Furthermore, it remains an open question how to determine the point S_e that is the projection of the point where the two folds of the surface \mathcal{S}_r meet for non-constant radius functions r (see Section 4.3.2). In this context, we have also left it open to identify the curve segments u_1 and u_2 that connect S_e to the two cusp points S_1 and S_2 . These curve segments border the threefold covered region of the projection of the swallowtail surface onto the plane (see Section 4.3.2). Similarly, there are other necessary aspects to accomplish the essential open task of adding an implementation for the case of strokes with singular boundary points to our algorithm presented in Chapter 5.

The implementation of the stroke model and mapping algorithm is already quite fast due to the GPU-based computations. However, achieving even faster computations would be a rewarding task. Faster calculations would improve the drawing experience, as it is only possible to draw and see the stroke appear at the same time if the user draws slowly.

The clarity of the code could be improved by reducing the number of required textures. In particular, in the case of self-intersection and, as proposed but not yet implemented, for regions with singular boundary points, many additional textures would be opened each time such a special case occurs. Reducing the number of textures involved would also reduce the time needed for the computations.

Another open task related to implementation is to revise the calculation of the global deviation of the outcome of our algorithm from conformality in the tile-based stroke implementation from Chapter 6. Currently, the global deviation value jumps to a higher value from time to time when the algorithm detects a self-intersection. We have identified that the jump is related to the global deviation value of the first part of the stroke.

In Chapter 2, our stroke model was based on a continuous, differentiable, regular curve γ . In Chapter 4, we interpolated the curve γ with cubic B-splines based on the registered discrete data points on the user-drawn path. Throughout the thesis, we have excluded the case that γ itself has singularities.

However, drawing a path with a cusp, as shown in Figure 7.1, can cause the curve γ to be singular. Integrating user-drawn paths with singularities into the model and algorithm would be an interesting extension of this work.

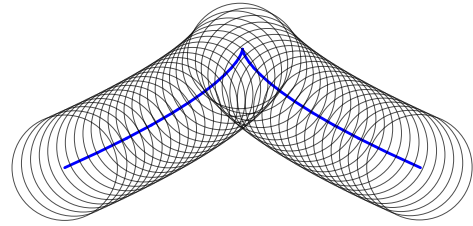


Figure 7.1: Curve γ with a cusp.

Besides the requirement that γ must be regular, it was also required that the curve drawn by the user must not be closed. It would be exciting to study how closed curves, and thus closed strokes, could be included in our model and algorithm. One difficulty would be to possibly change the interpolated curve in such a way that *almost* closing curves would also be treated as closing. Additionally, the potentially different radius values at both ends of γ would need to be adjusted to one another, which could affect not only the area in which the curve closes. Furthermore, the ornamental design would most likely need to be changed to match the closing stroke.

A related topic is blending of ornamental designs in the case of a self-intersecting stroke. So far, the different layers of the stroke are displayed on top of each other, as illustrated in Sections 4.3.1 and 5.4.1. Depending on the opacity of the stroke, only the last layer is visible, or the previous layers may show through. In any case, the artistic content of the layers remains separate, but continues seamlessly on the new part of the stroke. It would be interesting to develop an algorithm that merges the designs when the stroke passes over already covered parts of the drawing surface.

This is challenging, however, because the self-intersection area of the stroke can be very complex. Figure 7.2 shows examples of easy self-intersections and of cases that are most probably very hard to handle. The region covered twice can either be a rather simple quadrangle, as shown on the left, or an elongated, non-convex region, as shown on the right. If a stroke intersects itself multiple times, a region may even be covered more than twice.



Figure 7.2: Regions of a double cover of self-intersecting strokes.

If there are multiple self-intersections of a stroke, it would be desirable to add a condition to the implementation that the new part of the stroke alternately passes over and under the existing stroke, similar to the way knots and braids are handled.

List of Figures

1.1	Mapping an ornamental strip to a simulated pen stroke.	1
1.2	Finite strip consisting of $n = 4.5$ ornamental tiles T	2
1.3	A family of circles along a curve γ and the result of our algorithm.	3
1.4	A coordinate system within the stroke.	5
1.5	Testing the outcome of our algorithm for conformality.	6
1.6	Adjustment of folding regions in skeletal strokes [Ase10].	9
1.7	Exemplary results of synthesized pattern from [ZLL13, Figure 7].	10
1.8	Pattern strokes drawn with Adobe Illustrator’s pattern brush.	12
1.9	Decorative procedural growth from [MM12, Fig. 7].	13
2.1	Counterexamples of subsequent circles along a curve.	21
2.2	Pencils of circles defined by two circles.	23
2.3	The pressure waves of an airplane.	24
2.4	Examples for envelopes of families of different curves.	27
2.5	The boundary of a stroke consisting of the envelope curves and circular arcs.	29
2.6	Two intersecting circles $\mathcal{C}(t)$ and $\mathcal{C}(t_0)$	30
2.7	A change in the radius determines the direction of intersection.	31
2.8	A curve γ , its parallel $\gamma_{\parallel r_c}$ for $r_c = 6$ and its evolute ν	33
2.9	Derivation of the definition of E_d	35
2.10	Manifold $M = F^{-1}(0)$ and its projection $\pi(M) \in \mathbb{R}^2$	38
2.11	Orange manifold $M = F^{-1}(0)$ intersects blue $\partial M = \left(\frac{\partial F}{\partial t}\right)^{-1}(0)$ in red Σ	40
2.12	The ordinary cusp at zero of $4y^3 + 27x^2 = 0$	53
2.13	A set of parallels $\gamma_{\parallel r_c}$ of an ellipse γ	55
2.14	A swallowtail surface from different perspectives.	56
2.15	Sections of a swallowtail surface with planes for different values of r_c	57
2.16	Cuspidal edges of the swallowtail surface and the curve’s evolute.	58
2.17	The lifted red envelope curve for non-constant radius does not lie on \mathcal{S}	59
2.18	Surface \mathcal{S}_r for the stroke from Benchmark Example 2.1.10.	60
2.19	Projection of the surfaces \mathcal{S}_{r_c} (Example 2.6.10) and \mathcal{S}_r (Example 2.1.10).	61
2.20	Surface \mathcal{S}_{r_c} evolving for γ from Example 2.6.10 and $r_c = \frac{1}{4}$ rotated by $\frac{\pi}{2}$	61
3.1	Extended conformal map w determined by triples of boundary points.	64
3.2	Two ornamental strips with different lengths mapped to the same stroke.	65
3.3	Unique conformal map m of a quadrilateral to a rectangle.	66
3.4	Vertices of a rectangular ornamental strip.	69
3.5	The a - and b -sides of Q conformally mapped to R of side lengths a and b	69

List of Figures

3.6	Modifications on quadrilaterals and their vertices.	70
3.7	A rectangle and its conformal image under $f(w) = i\rho e^{-iw}$: a half annulus.	72
3.8	A stroke approximating a half annulus.	73
3.9	Conformal image of point z outside the domain by Schwarz reflection.	74
3.10	Analytic reflection along curve δ : $R_\delta(z) = w$	75
3.11	Circles of curvature for the inner envelope curve of an elliptic stroke.	76
4.1	Linear spline interpolation: linear segments visible in the stroke.	79
4.2	Hermite spline interpolated stroke with C^0 -continuous envelope.	81
4.3	Issues of cubic B-spline interpolation.	83
4.4	Cubic B-spline interpolated stroke with C^1 -continuous envelope curves.	84
4.5	Placing circles too far apart causes disjoint or wavy strokes.	84
4.6	The pixel grid underlying the stroke and the related quadrangulation.	86
4.7	Discrete analytic functions mapping a circle packing from the unit disk \mathbb{D} to circle packings with different combinatorics. [DS95, Figure 4]	92
4.8	A branching circle packing with branch cut. [DS95, Figure 8]	93
4.9	Detection of self-intersection of a stroke.	95
4.10	A stroke, its envelope curves and the second derivative of F	97
4.11	Stroke with boundary point of type $A_{\geq 3}$	97
4.12	Benchmark ellipse 2.1.9 with cusps and folded regions.	98
4.13	Benchmark Example 2.1.10 with cusps and folded regions.	99
5.1	The preimage region of a stroke in the rectangular strip S_n	105
5.2	The local map τ_d as initial map from the stroke to the strip.	106
5.3	Circular crescents are iteratively added to the stroke.	108
5.4	The visual result of the initialization of preimage coordinates in a stroke.	111
5.5	Force equilibrium of four springs attached at $f(q)$	112
5.6	The 16 neighbors of a pixel p with a distance of at most 2 pixels.	113
5.7	Error occurring when modulo calculations are neglected before averaging.	114
5.8	Reflection at circles instead of tangents to the envelope curve.	118
5.9	The ornament flowing to the right while averaging is active.	118
5.10	Adapted “reflection” for pixels outside the stroke near an envelope curve.	118
5.11	Interruption of pixel averaging when the drawing stops.	119
5.12	Rectangular grid of pixels around p_0 in the stroke.	120
5.13	Test for conformality while the stroke is drawn.	122
5.14	Results of the pixel averaging algorithm.	125
5.15	Textures containing the two parts of a self-intersecting stroke.	129
5.16	Strokes with self-intersection.	130
5.17	Self-intersecting strokes with visual and numeric test for local conformality.	131
5.18	Benchmark Examples 2.1.9 and 2.1.10 with cusps and special points.	131
5.19	Splitting a multiply folded stroke into three disjoint regions.	133
5.20	Erroneous detection of cusps and the related special points.	134
6.1	Simple example of creating visual output based on coordinates.	135
6.2	Drawback of using coordinate based rules: no MIP mapping.	136
6.3	Pressure-sensitive design and broken stars.	137
6.4	Four different textures of floral design.	140

6.5	Stroke with floral design varying with the stroke's width.	141
6.6	Stroke with floral design varying with the stroke's width.	142
6.7	Textures of braid design.	143
6.8	Stroke with variable braid design.	144
6.9	Braid tiles for the beginning and end of a stroke.	145
6.10	Drawing a stroke with special image tiles for the start and end of the stroke.	145
6.11	Four different textures with tangent Apollonian circles.	146
6.12	Strokes of different constant widths and variable design of tangent circles.	146
6.13	Strokes of constant width and design depending on their lengths.	147
6.14	Stroke with text and ornamentation.	147
7.1	Curve γ with a cusp.	149
7.2	Regions of a double cover of self-intersecting strokes.	149
A.1	Map from an ellipse E to an infinite strip S via the unit disk U	161
A.2	Conformal map of the unit disk to an infinite strip.	162
A.3	Map of the upper half plane H_+ to the rectangle R	163
A.4	Periodicity of sn_k with fundamental rectangle R_{sn} [Neh75, Fig.35].	164
A.5	Conformal map of an ellipse to the unit disk.	170
B.1	Basic logarithmic spirals.	171
B.2	Tangents including the same angle α with a line through the pole.	172
B.3	The poles and the tangent point of the spirals are collinear.	173
B.4	Symmetric double spirals for different parameters k	174
B.5	Two spirals ℓ_1 and ℓ_2 with two points of contact T_1 and T_2	174
B.6	Intertwined tangent spirals for different parameters k	175
B.7	Calculating the angle between $P_2 - P_1$ and $\ell_1(0) - P_1$	176
B.8	The lines m_1 and m_2 are reflections with respect to the tangent t	177
B.9	T lies between $\ell_1(0)$ and $\ell_1(\theta)$ on $\text{join}(P_1, P_2)$	178
B.10	Tangent spirals placed inside a stroke defined by a family of circles.	179
B.11	User interface to select different types of tangential spirals.	180
B.12	Strokes with different types of tangential spirals.	180
B.13	Definition of a new spiral in the stroke.	182

Bibliography

- [And16] John D Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill Education, sixth edition, 2016.
- [Ase10] Paul J Asente. Folding Avoidance in Skeletal Strokes. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, pages 33–40, 2010.
- [Bea85] AF Beardon. Conformal Automorphisms of Plane Domains. *Journal of the London Mathematical Society*, 2(2):245–253, 1985.
- [BG92] James W Bruce and Peter J Giblin. *Curves and Singularities: a Geometrical Introduction to Singularity Theory*. Cambridge university press, 1992.
- [BGT20] Kelly Bickel, Pamela Gorkin, and Trung Tran. Applications of Envelopes. *Complex Analysis and its Synergies*, 6(1):1–14, 2020.
- [Böl13] Ludwig Bölkow. *Ein Jahrhundert Flugzeuge: Geschichte und Technik des Fliegens*. Springer-Verlag, 2013.
- [BPS15] Alexander I Bobenko, Ulrich Pinkall, and Boris A Springborn. Discrete Conformal Maps and Ideal Hyperbolic Polyhedra. *Geometry & Topology*, 19(4):2155–2215, 2015.
- [BSS16] Alexander I Bobenko, Stefan Sechelmann, and Boris A Springborn. *Discrete Conformal Maps: Boundary Value Problems, Circle Domains, Fuchsian and Schottky Uniformization*. Springer Berlin Heidelberg, 2016.
- [Car98] Constantin Carathéodory. *Conformal Representation*. Courier Corporation, 1998.
- [CG17] Renjie Chen and Craig Gotsman. Approximating Planar Conformal Maps Using Regular Polygonal Meshes. In *Computer Graphics Forum*, volume 36, pages 629–642. Wiley Online Library, 2017.
- [Con19] Conrad. Answer to *Quasiconformal Mappings - Definition of Modulus of a 'Quadrilateral'*. <https://math.stackexchange.com/questions/3393864/quasiconformal-mappings-definition-of-modulus-of-a-quadrilateral>, 15.10.2019. Online, StackExchange.

- [Cox69] Harold SM Coxeter. *Introduction to Geometry*. John Wiley & Sons, Inc., second edition, 1969.
- [Cra19] Keenan Crane. Conformal Geometry of Simplicial Surfaces. In *An Excursion Through Discrete Differential Geometry, Proceedings of Symposia in Applied Mathematics*, pages 59–102, 2019.
- [CW15] Renjie Chen and Ofir Weber. Bounded Distortion Harmonic Mappings in the Plane. *ACM Transactions on Graphics (TOG)*, 34(4):1–12, 2015.
- [Dap18] Dap. Answer to *Conformal Mapping from Square to Rectangle Preserving Corners*. <https://math.stackexchange.com/questions/2709809/conformal-mapping-from-square-to-rectangle-preserving-corners>, 28.03.2018. Online, StackExchange.
- [Del94] Thomas K Delillo. The Accuracy of Numerical Conformal Mapping Methods: a Survey of Examples and Results. *SIAM Journal on Numerical Analysis*, 31(3):788–812, 1994.
- [DS95] Tomasz Dubejko and Kenneth Stephenson. Circle Packing: Experiments in Discrete Analytic Function Theory. *Experimental Mathematics*, 4(4):307–348, 1995.
- [DT02] Tobin A Driscoll and Lloyd N Trefethen. *Schwarz-Christoffel Mapping*, volume 8. Cambridge University Press, 2002.
- [DV98] Tobin A Driscoll and Stephen A Vavasis. Numerical Conformal Mapping Using Cross-ratios and Delaunay Triangulation. *SIAM Journal on Scientific Computing*, 19(6):1783–1803, 1998.
- [Esk22] Hossein Eskandari. Strictly Conformal Transformation Optics for Directivity Enhancement and Unidirectional Cloaking of a Cylindrical Wire Antenna. *Scientific Reports*, 12(1):16278, 2022.
- [Far02] Rida T Farouki. *Pythagorean-hodograph Curves*. Springer, 2002.
- [Fow20] Ralph H Fowler. *The Elementary Differential Geometry of Plane Curves*. University Press, 1920.
- [FPS01] Maria I Falcão, Nicolas Papamichael, and Nikos S Stylianopoulos. Approximating the Conformal Maps of Elongated Quadrilaterals by Domain Decomposition. *Constructive Approximation*, 17:589–617, 2001.
- [GHY⁺15] Chao Guo, Zengxuan Hou, Guangqing Yang, Shuanzhu Zheng, et al. The Simulation of the Brush Stroke Based on Force Feedback Technology. *Mathematical Problems in Engineering*, 2015.
- [GLSW18] Xianfeng D Gu, Feng Luo, Jian Sun, and Tianqi Wu. A Discrete Uniformization Theorem for Polyhedral Surfaces. *Journal of Differential Geometry*, 109(2):223–256, 2018.

Bibliography

- [GNH⁺17] Yanzhou Gong, Ziqiang Ni, Weixing Huang, Jian Wang, and Guigang Zhang. A Real-time Chinese Calligraphy Creation System. In *2017 IEEE International Symposium on Multimedia (ISM)*, pages 536–542. IEEE, 2017.
- [GS16] Paul M Gauthier and Fatemeh Sharifi. The Carathéodory Reflection Principle and Osgood–Carathéodory Theorem on Riemann Surfaces. *Canadian Mathematical Bulletin*, 59(4):776–793, 2016.
- [Hen93] Peter Henrici. *Applied and Computational Complex Analysis, Volume 3: Discrete Fourier Analysis, Cauchy Integrals, Construction of Conformal Maps, Univalent Functions*, volume 41. John Wiley & Sons, Inc., 1993.
- [HL94] Siu C Hsu and Irene HH Lee. Drawing and Animation Using Skeletal Strokes. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 109–118, 1994.
- [HLW93] Siu C Hsu, Irene HH Lee, and Neil E Wiseman. Skeletal Strokes. In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, pages 197–206, 1993.
- [Hol22] Freya Holmér. *The Continuity of Splines*. <https://www.youtube.com/watch?v=jvPPXbo87ds>, 07.12.2022. Online, YouTube.
- [HQR13] Harri Hakula, Tri Quach, and Antti Rasila. Conjugate Function Method for Numerical Conformal Mappings. *Journal of Computational and Applied Mathematics*, 237(1):340–353, 2013.
- [Inc19] Adobe Inc. *Adobe Illustrator Help*. https://helpx.adobe.com/pdf/illustrator_reference.pdf, 2019.
- [Kai23] Surinder PS Kainth. *A Comprehensive Textbook on Metric Spaces*. Springer Nature, 2023.
- [KCK19] Eryk Kopczyński and Dorota Celińska-Kopczyńska. Conformal Mappings of the Hyperbolic Plane to Arbitrary Shapes. In *Proceedings of Bridges 2019: Mathematics, Art, Music, Architecture, Education, Culture*, pages 91–98, 2019.
- [Kno99] Gary D Knott. *Interpolating Cubic Splines*, volume 18. Springer Science & Business Media, 1999.
- [Kob52] Hermann Kober. *Dictionary of Conformal Representations*. Dover Publications, 1952.
- [Kön13] Konrad Königsberger. *Analysis 2*. Springer-Verlag, 2013.
- [KP02] Steven G Krantz and Harold R Parks. *The Implicit Function Theorem: History, Theory, and Applications*. Springer Science & Business Media, 2002.
- [KS23] Julia Kowalczyk and Jannik Steinmeier. Private communication, 2023. Technical University Munich.

- [Lan13] Serge Lang. *Complex Analysis*, volume 103. Springer Science & Business Media, 2013.
- [LBW⁺14] Jingwan Lu, Connelly Barnes, Connie Wan, Paul Asente, Radomir Mech, and Adam Finkelstein. Decobrush: Drawing Structured Decorative Patterns by Example. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014.
- [LSW22] Feng Luo, Jian Sun, and Tianqi Wu. Discrete Conformal Geometry of Polyhedral Surfaces and its Convergence. *Geometry & Topology*, 26(3):937–987, 2022.
- [LV73] Olli Lehto and Kaarlo I Virtanen. *Quasiconformal Mappings in the Plane*, volume 126. Citeseer, 1973.
- [M⁺17] Aaron Montag et al. *CindyGL Tutorial*. <https://cindyjs.org/docs/cindygltutorial/>, last edited 2017. Online.
- [MM12] Radomir Mech and Gavin Miller. The Deco Framework for Interactive Procedural Modeling. *Journal of Computer Graphics Techniques (JCGT)*, 1(1):43–99, 2012.
- [Mon20] Aaron Montag. *Domain Parallel Machines*. PhD thesis, Technische Universität München, 2020.
- [MR07] Donald E Marshall and Steffen Rohde. Convergence of a Variant of the Zipper Algorithm for Conformal Mapping. *SIAM Journal on Numerical Analysis*, 45(6):2577–2609, 2007.
- [MRG20] Aaron Montag and Jürgen Richter-Gebert. Private communication, 2020. Technical University Munich.
- [Nee23] Tristan Needham. *Visual Complex Analysis*. Oxford University Press, 2023.
- [Neh75] Zeev Nehari. *Conformal Mapping*. Dover Publications, 1975.
- [NRR⁺22] Mohamed Nasser, Oona Rainio, Antti Rasila, Matti Vuorinen, Terry Wallace, Hang Yu, and Xiaohui Zhang. Polycircular Domains, Numerical Conformal Mappings, and Moduli of Quadrilaterals. *Advances in Computational Mathematics*, 48(5):58, 2022.
- [OLG⁺07] John D Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E Lefohn, and Timothy J Purcell. A Survey of General-purpose Computation on Graphics Hardware. In *Computer Graphics Forum*, volume 26, pages 80–113. Wiley Online Library, 2007.
- [Pal75] Bruce Palka. Conformal Mappings and the Distortion of Moduli of Rectangles. *Proceedings of the American Mathematical Society*, 50(1):155–161, 1975.
- [Pen06] Roger Penrose. *The Road to Reality*. Random house, 2006.

Bibliography

- [Per13] Geoffrey J Pert. *Introductory Fluid Mechanics for Physicists and Mathematicians*. John Wiley & Sons, 2013.
- [PH13] Przemyslaw Prusinkiewicz and James Hanan. *Lindenmayer Systems, Fractals, and Plants*, volume 79. Springer Science & Business Media, 2013.
- [Pol23a] Lena Polke. Parallelized pixel averaging for a real-time calligraphic pen. In *Bridges 2023 Conference Proceedings*, pages 165–172. Tessellations Publishing, 2023.
- [Pol23b] Lena Polke. *GitHub Repository: Parallelized-Approaches-to-Conformal-Mappings*. <https://github.com/LenaPolke/Parallelized-Approaches-to-Conformal-Mappings>, 2023.
- [PRG21] Lena Polke and Jürgen Richter-Gebert. Real-time Ornamental Calligraphic Pens. In *Bridges 2021 Conference Proceedings*, pages 141–148. Tessellations Publishing, 2021.
- [PS10] Nicolas Papamichael and Nikos S Stylianopoulos. *Numerical Conformal Mapping: Domain Decomposition and the Mapping of Quadrilaterals*. World Scientific, 2010.
- [Rei23] Tim Reinhardt. Approximation of Conformal Mappings on the GPU. Master’s thesis, Technische Universität München, 2023.
- [RG11] Jürgen Richter-Gebert. *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry*. Springer, 2011.
- [RG21] Jürgen Richter-Gebert. Private communication, 2021. Technical University Munich.
- [RGK23a] Jürgen Richter-Gebert and Ulrich Kortenkamp. Cinderella. The Interactive Geometry Software Cinderella. <https://cinderella.de/tiki-index.php>, access: 2023. last modification: 2013.
- [RGK23b] Jürgen Richter-Gebert and Ulrich Kortenkamp. CindyJS. <https://cindyjs.org>, access: 2023.
- [RS87] Burt Rodin and Dennis Sullivan. The Convergence of Circle Packings to the Riemann Mapping. *Journal of Differential Geometry*, 26(2):349–360, 1987.
- [Rut18] John W Rutter. *Geometry of Curves*. Chapman and Hall/CRC, 2018.
- [SBC16] Aviv Segall and Mirela Ben-Chen. Iterative Closest Conformal Maps Between Planar Domains. In *Computer Graphics Forum*, volume 35, pages 33–40. Wiley Online Library, 2016.
- [Ser68] Joseph A Serret. *Handbuch der Höheren Algebra*, volume 1. Teubner, 1868.
- [SKR19] Jungpil Shin, Cheol M Kim, and Md Abdur Rahim. Simulating Oriental Brush Character Considered With Aerial Action of Pen Tablet. *Multimedia Tools and Applications*, 78:19341–19359, 2019.

- [SM03] Endre Süli and David F Mayers. *An Introduction to Numerical Analysis*. Cambridge university press, 2003.
- [Ste99] Kenneth Stephenson. The Approximation of Conformal Structures via Circle Packing. In *Computational Methods and Function Theory 1997: Proceedings of the Third CMFT Conference*, pages 551–582. World Scientific, 1999.
- [Ste03] Kenneth Stephenson. Circle Packing: a Mathematical Tale. *Notices of the AMS*, 50(11):1376–1388, 2003.
- [Ste05] Kenneth Stephenson. *Introduction to Circle Packing: The Theory of Discrete Analytic Functions*. Cambridge University Press, 2005.
- [Swa11] David Swart. Warping Pictures Nicely. In *Proceedings of Bridges 2011: Mathematics, Music, Art, Architecture, Culture*, pages 303–310, 2011.
- [SWG15] Jian Sun, Tianqi Wu, Xianfeng Gu, and Feng Luo. Discrete Conformal Deformation: Algorithm and Experiments. *SIAM Journal on Imaging Sciences*, 8(3):1421–1456, 2015.
- [Tho75] René Thom. *Structural Stability and Morphogenesis, An Outline of a General Theory of Models*. W. A. Benjamin, Inc., 1975.
- [Tre20] Lloyd N Trefethen. Numerical Conformal Mapping with Rational Functions. *Computational Methods and Function Theory*, 20:369–387, 2020.
- [VGdF07] Luiz Velho, Jonas Gomes, and Luiz H de Figueiredo. *Implicit Objects in Computer Graphics*. Springer Science & Business Media, 2007.
- [WCW⁺18] Rundong Wu, Zhili Chen, Zhaowen Wang, Jimei Yang, and Steve Marschner. Brush Stroke Synthesis with a Generative Adversarial Network Driven by Physically Based Simulation. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pages 1–10, 2018.
- [Wei23a] Eric W Weisstein. Jacobi Elliptic Functions. From *MathWorld* – A Wolfram Web Source. <https://mathworld.wolfram.com/JacobiEllipticFunctions.html>, 01.03.2023. Online, Wolfram MathWorld.
- [Wei23b] Eric W Weisstein. Jacobi Theta Functions. From *MathWorld* – A Wolfram Web Source. <https://mathworld.wolfram.com/JacobiThetaFunctions.html>, 01.03.2023. Online, Wolfram MathWorld.
- [Wer23] Bernhard O Werner. Private communication, 2023. Technical University Munich.
- [WG10] Ofir Weber and Craig Gotsman. Controllable Conformal Maps for Shape Deformation and Interpolation. In *ACM SIGGRAPH 2010 Papers*, pages 1–11. 2010.

Bibliography

- [WI00] Helena TF Wong and Horace HS Ip. Virtual Brush: a Model-based Synthesis of Chinese Calligraphy. *Computers & Graphics*, 24(1):99–113, 2000.
- [WR23] Inc. Wolfram Research. Mathematica. <https://www.wolfram.com/mathematica/>, access: 2023. Champaign, IL, 2023.
- [WZS98] Michael T Wong, Douglas E Zongker, and David H Salesin. Computer-generated Floral Ornament. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 423–434, 1998.
- [XLTP03] Songhua Xu, Francis CM Lau, Feng Tang, and Yunhe Pan. Advanced Design for a Realistic Virtual Brush. In *Computer Graphics Forum*, volume 22, pages 533–542. Wiley Online Library, 2003.
- [Zee77] Erik C Zeeman. *Catastrophe Theory, Selected Papers 1972-1977*. Addison-Wesley Publishing Company, Inc., 1977.
- [Zee79] Erik C Zeeman. Catastrophe Theory. In Werner Güttinger and Horst Eike-meier, editors, *Structural Stability in Physics*, pages 12–22, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg.
- [ZLL13] Shizhe Zhou, Anass Lasram, and Sylvain Lefebvre. By-example Synthesis of Curvilinear Structured Patterns. In *Computer Graphics Forum*, volume 32, pages 355–360. Wiley Online Library, 2013.

A | Conformal Map of an Ellipse to an Infinite Strip

In this chapter, we calculate a conformal map from the interior of an ellipse E to an infinite strip S . For this, the ellipse E is centered at the origin and is given by the equation

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,$$

where $a, b > 0$ are the lengths of the semi-major and -minor axes. The infinite strip S is defined as $-\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$. To determine the exact map, we concatenate a map w from the ellipse to the unit disk U given by $|z| = |x + iy| \leq 1$ and a map m from the unit disk to the strip as shown in Figure A.1.

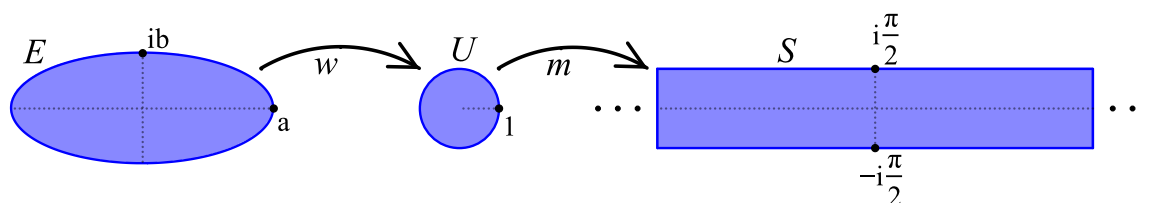


Figure A.1: Map from an ellipse E to an infinite strip S via the unit disk U .

The second map $m: U \rightarrow S$ can be computed by composition of the following three maps. First, the Möbius transformation $m_1: U \rightarrow H_+$ maps the unit disk to the upper half plane $H_+ = \{z = x + iy \mid y \geq 0\}$ by

$$m_1: z \mapsto \frac{i(z+1)}{1-z}.$$

This can be easily verified by checking that $m(i) = -1$, $m(-1) = 0$, and $m(-i) = 1$. A Möbius transformation is uniquely defined by three pairs of preimage and image points, and it maps circles/lines to circles/lines. Hence we see that m maps the unit circle $|z| = 1$ to the horizontal line $y = 0$. The interior of U is mapped to the upper half plane and not the lower one, as m_1 maps the origin to i .

Second, the upper half plane H_+ is rotated by -90° and mapped onto the right side of the complex plane $\mathbb{C}_{x \geq 0} := \{z = x + iy \in \mathbb{C} \mid 0 \leq x, y \in \mathbb{R}\}$:

$$m_2: H_+ \rightarrow \mathbb{C}_{x \geq 0} \quad z \mapsto z \cdot e^{-i\frac{\pi}{2}} = -i \cdot z.$$

Finally, the complex logarithm maps the right half-plane onto the strip S :

$$m_3: \mathbb{C}_{x \geq 0} \rightarrow S, \quad z \mapsto \log(z).$$

This is verified by writing $z \in \mathbb{C}_{x \geq 0}$ in polar coordinates as $z = re^{i\varphi}$ for some real radius $r \geq 0$ and some real angle $\varphi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. Applying m_3 yields $m_3(z) = \log(re^{i\varphi}) = \log(r) + i\varphi$, where $\log(r) \in (-\infty, \infty)$ since $r \geq 0$ and $\varphi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. Hence, $m_3(z)$ is located in S .

In total, the unit disk U is mapped to the infinite strip S by

$$m: U \rightarrow S, \quad z \mapsto m_3 \circ m_2 \circ m_1(z) = \log\left(\frac{z+1}{1-z}\right).$$

This map m is made visible by using reverse pixel lookup, which is explained in Section 5.1.3. For this, the infinite strip is, for example, filled with a checkerboard pattern as depicted in Figure A.2 on the right. Then, for each pixel p within the unit disk on the left of Figure A.2, we calculate the image point $m(p) = p'$ in S , retrieve the color $c_{p'} = (r, g, b)$ at the image pixel p' from the strip, and display it at the preimage pixel p .

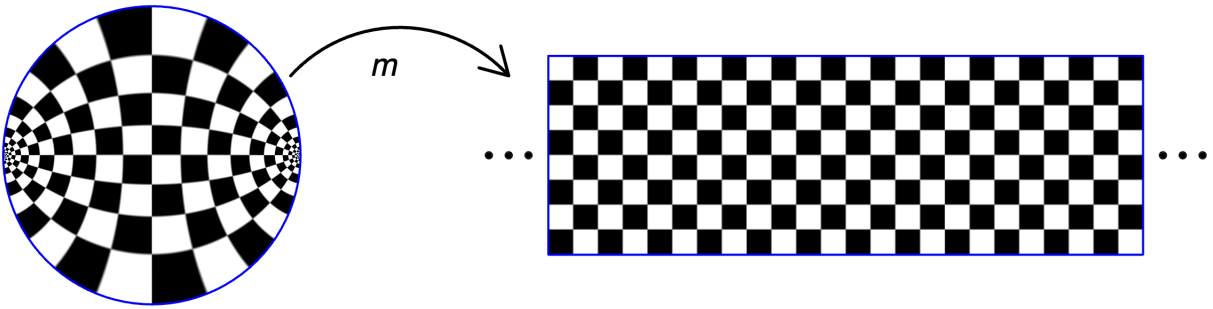


Figure A.2: Conformal map of the unit disk to an infinite strip.

Mapping an ellipse E to the unit disk U is more challenging. An explicit formula for this can be found in the work of Kober [Kob52, p.177]:

$$w: E \rightarrow U$$

$$z \mapsto \sqrt{k} \cdot \operatorname{sn}\left(\frac{2K}{\pi} \sin^{-1}\left(\frac{z}{\sqrt{a^2 - b^2}}\right), k\right) \quad (\text{A.1})$$

where sn is the *Jacobi elliptic sine function* and

$$K = \int_0^{\frac{\pi}{2}} \frac{1}{1 - k^2 \sin^2 \varphi} d\varphi$$

$$k = \left(\frac{\theta_2(\tau)}{\theta_3(\tau)}\right)^2$$

$$\tau = \frac{2i}{\pi} \log\left(\frac{a+b}{a-b}\right) \sim e^{i\pi\tau} = \left(\frac{a-b}{a+b}\right)^2.$$

Although the formula for w in Equation (A.1) is given, it is not obvious how to determine the coordinates of $w(z) \in U$ for $z \in E$. Section A.1 is dedicated to the derivation of the formula for w as given in Equation (A.1). In Section A.2, we will examine how to compute w .

A.1 Jacobi elliptic sine function: map from an ellipse to the unit disk

In the equation for w in (A.1), the Jacobi elliptic sine function sn is the core component. To examine its characteristics, we follow the argumentation of Nehari [Neh75, p.280 ff.].

We consider the map F of the upper half plane H_+ to a rectangle R with corners at $-K, K, K + iK', -K + iK'$ where $K, K' \in \mathbb{R}$. For a parameter $k \in \mathbb{R}$, F is defined as

$$F: H_+ \rightarrow R, \quad x \mapsto z = F(x) = \int_0^x \frac{1}{\sqrt{(1-\xi^2)(1-k^2\xi^2)}} d\xi.$$

This is a conformal Schwarz-Christoffel mapping (SCmap for short) of the upper half plane H_+ to the rectangle R . It maps the following points on the real axis to the corners of the rectangles, which is also shown in Figure A.3:

$$-1 \mapsto -K, \quad 1 \mapsto K, \quad \frac{1}{k} \mapsto K + iK', \quad -\frac{1}{k} \mapsto -K + iK'.$$

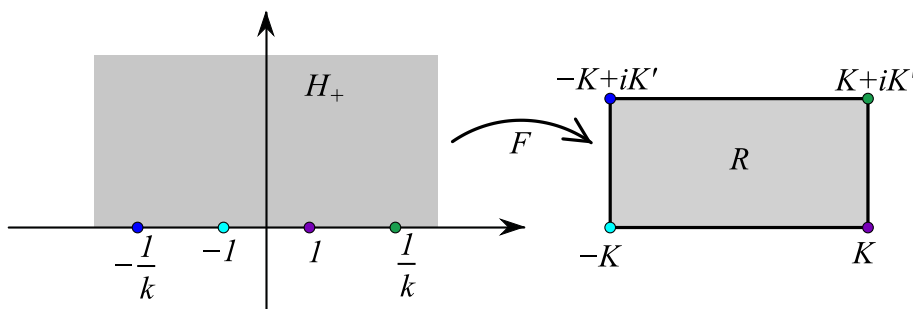


Figure A.3: Map of the upper half plane H_+ to the rectangle R .

Following Driscoll and Trefethen [DT02, Section 2.5] or Nehari [Neh75, Chapter V, Section 6], the SCmap F can be derived from the general formula for the conformal SCmap of the upper half-plane to a polygon with n corners. Let a_i for $i \in \{1, \dots, n\}$ be the points on the real line $y = 0$ that are supposed to be mapped to the corners of the n -gon. Let $\mu_i\pi$ be the interior angle that the polygon has at corner i . Then the conformal map is given by

$$f(x) = A + C \int_0^x \frac{1}{\prod_i (\xi - a_i)^{1-\mu_i}} d\xi$$

for some constants $A, C \in \mathbb{C}$ [DT02, Theorem 1.1], [Neh75, p.192]. If we set $n = 4$, take $-\frac{1}{k}$, -1 , 1 , and $\frac{1}{k}$ as points a_i , and assume all interior angles to be right angles, i.e., $\mu_i\pi = \frac{\pi}{2}$, we get equality of f and F for the constants $A = 0$ and $C = 1$:

$$f(x) = \int_0^x \frac{1}{\sqrt{(\xi+1)(\xi-1)(\xi-\frac{1}{k})(\xi+\frac{1}{k})}} d\xi = F(x).$$

The resulting rectangle R is symmetric about the vertical line $x = 0$. The inverse of F is an analytic function, which is called the *Jacobi elliptic sine function*

$$\operatorname{sn}_k(z) = \operatorname{sn}(z, k) = F^{-1}(z) = x$$

that depends on $k \in (0, 1)$. It conformally maps a rectangle to the upper half plane. However, instead of k , one usually uses

$$\tau = \frac{iK'}{K} \quad \text{or} \quad q = e^{i\pi\tau} = e^{-\pi\frac{K'}{K}} \quad (\text{A.2})$$

where k is uniquely defined by τ and q . To determine these unknowns, we require the equations

$$K = \int_0^1 \frac{1}{\sqrt{(1-t^2)(1-k^2t^2)}} dt \quad \text{and} \quad K' = \int_0^1 \frac{1}{\sqrt{(1-t^2)(1-k'^2t^2)}} dt$$

where $k' = \sqrt{1-k^2}$ and thus $K'(k) = K(k')$.

Note A.1.1 *The name Jacobi elliptic sine function comes from the analogies to the Euclidean sine function \sin , which is a special case of the elliptic sn_k for $k \rightarrow \infty$. It is called an elliptic function because the integral F was first explored in the context of finding the length of the arc of an ellipse.*

Like the sine function, also the Jacobi elliptic sn_k is periodic; it has even two periods. See Figure A.4 for a visualization of the rectangles in the following explanations. The rectangle R has corners at $-K$, K , $K+iK'$, and $-K+iK'$ as defined above. It is mapped to the upper half plane by sn_k . Reflection of R at $y = iK'$ results in a rectangle R_1 , which in turn is mapped to the lower half plane. A further reflection at $y = 2iK'$ results in a rectangle which is again mapped to the upper half plane. Hence, one period is $2iK'$ and it holds $\text{sn}(z + 2iK', k) = \text{sn}(z, k)$. The same happens when R is reflected at $x = K$. The resulting rectangle R_2 is then mapped to the lower half plane. Reflecting R_2 at $x = 3K$ likewise maps the resulting rectangle to the upper half plane identically to R . Thus, the second period is $4K$, i.e., it holds $\text{sn}(z + 4K, k) = \text{sn}(z, k)$. The rectangle

$$R_{\text{sn}} = \{z = x + iy \in \mathbb{C} \mid -K \leq x < 3K, 0 \leq y < 2iK'\}$$

is called fundamental rectangle and covers the plane twice when it is mapped by the Jacobi elliptic sine function sn_k .

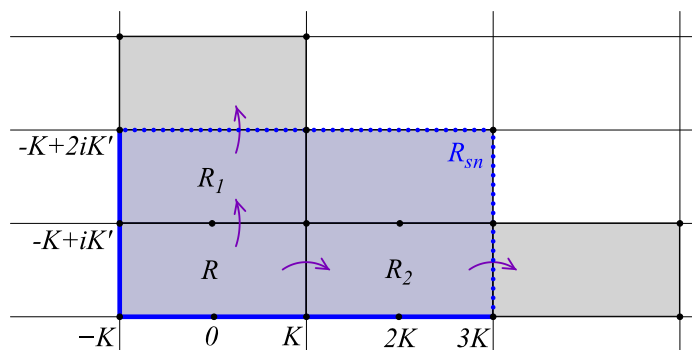


Figure A.4: Periodicity of sn_k with fundamental rectangle R_{sn} [Neh75, Fig.35].

Within the fundamental rectangle R_{sn} , only two values are mapped to zero: 0 and $2K$. Moreover, there are only two finite singularities at iK' and $2K + iK'$, which are simple

A.1. Jacobi elliptic sine function: map from an ellipse to the unit disk

poles. Due to periodicity, global zeros occur at $z = 2nK + 2imK'$ and global poles occur at $z = 2nK + (2m + 1)iK'$ for $n, m \in \mathbb{Z}$. The global poles and zeros are used to construct an infinite product that defines sn . For this, we consider the function

$$g(z) := \xi \cdot \frac{\prod_{m=0}^{\infty} (1 - q^{2m}\xi^{-2}) \cdot \prod_{m=1}^{\infty} (1 - q^{2m}\xi^2)}{\prod_{m=0}^{\infty} (1 - q^{2m+1}\xi^{-2}) \cdot \prod_{m=0}^{\infty} (1 - q^{2m+1}\xi^2)}$$

for $\xi = e^{\frac{\pi iz}{2K}}$ and $q = e^{-\pi \frac{K'}{K}}$ as in Equation (A.2). Since q is in $(0, 1)$, the series $\sum_m q^m$ converges. Hence, the four products in both the numerator and denominator of the function g converge absolutely since $\sum_{j=1}^n |a_j| \leq \prod_{j=1}^n (1 + |a_j|)$.

We investigate the properties of an exemplary product $Q = \prod_{m=1}^{\infty} (1 - q^m \xi)$. Taking the logarithm gives $\log Q = \sum_{m=1}^{\infty} \log(1 - q^m \xi)$ which converges if $\log(1 - q^m \xi) \xrightarrow{m \rightarrow \infty} 0$. This is true if $(1 - q^m \xi) \xrightarrow{m \rightarrow \infty} 1$ which holds since $q \in (0, 1)$. Therefore, Q is convergent and it even converges uniformly which makes it a regular analytic function [Neh75, p.96].

We can conclude that g is a regular analytic function at all finite points where the denominator does not vanish. The zeros of g are those points where the numerator vanishes:

$$\begin{aligned} 1 - q^{2m}\xi^{-2} = 0 &\Leftrightarrow q^{2m}\xi^{-2} = 1 &\Leftrightarrow e^{-\frac{2\pi m K'}{K} - \frac{\pi iz}{K}} = 0 &\quad \text{for } m \geq 0 \\ 1 - q^{2m}\xi^2 = 0 &\Leftrightarrow q^{2m}\xi^2 = 1 &\Leftrightarrow e^{-\frac{2\pi m K'}{K} + \frac{\pi iz}{K}} = 0 &\quad \text{for } m \geq 1 \end{aligned}$$

Taking the logarithm gives

$$\begin{aligned} -\frac{2\pi m K'}{K} - \frac{\pi iz}{K} &= 2\pi in &\quad \text{for } m \geq 0, n \in \mathbb{Z} \\ -\frac{2\pi m K'}{K} + \frac{\pi iz}{K} &= 2\pi in &\quad \text{for } m \geq 1, n \in \mathbb{Z} \end{aligned}$$

and thus $g(z) = 0 \Leftrightarrow z = 2nK + 2imK'$ for $n, m \in \mathbb{Z}$, which correspond exactly to the zeros of sn_k . Analogously, it can be shown that the poles of g and sn_k coincide.

Next, we show that g has the same double periodicity with periods $4K$ and $2iK'$ as sn_k . We observe that the variable z only occurs inside ξ , and it holds that

$$\xi = e^{\frac{\pi i}{2K}(z+4K)} = e^{\frac{\pi i}{2K}z} e^{2\pi i} = e^{\frac{\pi i}{2K}z} \tag{A.3}$$

$$\xi = e^{\frac{\pi i}{2K}(z+2iK')} = e^{\frac{\pi i}{2K}z} e^{-\frac{\pi K'}{K}} = q e^{\frac{\pi i}{2K}z} \tag{A.4}$$

From Equation (A.3), it is evident that one periodicity is in $4K$. To get the period of $2iK'$, we insert Equation (A.4) in g and get

$$g(z + 2iK') = q \cdot g(z) \frac{(1 - q^{-2}\xi^{-2})(1 - q\xi^2)}{(1 - q^2\xi^2)(1 - q^{-1}\xi^{-2})} = g(z).$$

This completes the proof that g and sn_k have the same periods.

To show equality of g and sn_k , we define the quotient $c(z) := \frac{\text{sn}(z,k)}{g(z)}$. As both the numerator and denominator have identical poles and zeros, they cancel, and c is regular

Appendix A. Conformal Map of an Ellipse to an Infinite Strip

at all finite points in the z -plane. Additionally, since both the numerator and denominator have the same periods, also c is periodic with periods $4K$ and $2iK'$ and has the same fundamental rectangle. Within the closure of this rectangle, c is bounded, which consequently also applies to the whole z -plane. It follows from the Theorem of Liouville [Neh75, p.117] that c is constant. Hence, the Jacobi elliptic sine function sn and g differ only by a constant $c \in \mathbb{C}$:

$$\text{sn}(z, k) = c \cdot g(z).$$

To explicitly determine the value of c , we insert some known values to sn and ξ :

$$\text{sn}(K, k) = 1, \quad \xi(K) = e^{\frac{\pi i K}{2K}} = i \quad \Rightarrow \quad 1 = 2ic \prod_{n=1}^{\infty} \left(\frac{1 + q^{2n}}{1 + q^{2n-1}} \right)^2 \quad (\text{A.5})$$

$$\text{sn}(K + iK', k) = \frac{1}{k}, \quad \xi(K + iK') = i\sqrt{q} \quad \Rightarrow \quad \frac{1}{k} = \frac{ic}{2\sqrt{q}} \prod_{n=1}^{\infty} \left(\frac{1 + q^{2n-1}}{1 + q^{2n}} \right)^2 \quad (\text{A.6})$$

Bringing together these two identities yields

$$\left(\frac{(\text{A.5})}{(\text{A.6})} \right)^2 : \quad k^2 = 16q \prod_{n=1}^{\infty} \left(\frac{1 + q^{2n}}{1 + q^{2n-1}} \right)^8. \quad (\text{A.7})$$

Since $q > 0$, we know by (A.5) that $ic > 0$ and, finally, that

$$c = -i \frac{\sqrt[4]{q}}{\sqrt{k}}.$$

Hence, the Jacobi elliptic sine function can be written in terms of the infinite product g :

$$\text{sn}(z, k) = -i \frac{\sqrt[4]{q}}{\sqrt{k}} \xi \frac{\prod_{m=0}^{\infty} (1 - q^{2m} \xi^{-2}) \prod_{m=1}^{\infty} (1 - q^{2m} \xi^2)}{\prod_{m=0}^{\infty} (1 - q^{2m+1} \xi^{-2}) \prod_{m=0}^{\infty} (1 - q^{2m+1} \xi^2)} \quad (\text{A.8})$$

for $\xi = e^{\frac{\pi iz}{2K}}$, $q = e^{-\pi \frac{K'}{K}}$ and k as the square root of Equation (A.7). Since k and q are dependent of each other, it is also common to write $\text{sn}(z, q)$.

Using the formula for sn in Equation (A.8) and following Nehari [Neh75, p.295 f.], we derive the map w from Equation (A.1) which maps an ellipse E to the unit disk U . For this, we assume that the ellipse E has foci ± 1 and semi-axes $a = \cosh(\zeta)$, $b = \sinh(\zeta)$ for some $\zeta > 0$. However, the results can be generalized for any ellipses. Furthermore, we assume that w maps the origin to itself and that its derivative at zero is positive. This is possible due to the Riemann mapping theorem, as stated by Nehari [Neh75, p.175]: the conformal map of a simply connected planar domain that is not the whole plane is unique if one defines a point to map to the origin and the derivative at that point to be positive. As a consequence of $w(0) = 0$ and $w'(0) > 0$, w maps the upper half of the ellipse to the upper half of the unit disk.

A.1. Jacobi elliptic sine function: map from an ellipse to the unit disk

The complex sine function \sin maps the rectangle

$$R_{\sin} := \{(x, y) \in \mathbb{R}^2 \mid -\frac{\pi}{2} < \operatorname{Re}(z) < \frac{\pi}{2}, 0 < \operatorname{Im}(z) < \zeta\}$$

to the upper half of the ellipse E [Neh75, Chap. VI, Sec. 2]. It holds that

$$\sin\left(-\frac{\pi}{2}\right) = -1, \quad \sin\left(\frac{\pi}{2}\right) = 1, \quad \sin\left(\frac{\pi}{2} + i\zeta\right) = \cosh(\zeta), \quad \sin\left(-\frac{\pi}{2} + i\zeta\right) = -\cosh(\zeta).$$

Thus, when given w , it holds that $w(\sin(z))$ maps the rectangle R_{\sin} to the upper half of the unit circle. By concatenation with a simple Möbius transformation $t(z) = \frac{2z}{1+z^2}$ that maps the half unit disk to the upper half plane, it holds that $t \circ w \circ \sin(z)$ maps R_{\sin} to H_+ . Additionally, for $\alpha = w(1)$ it holds

$$-\frac{\pi}{2} \mapsto -\frac{2\alpha}{1+\alpha^2}, \quad \frac{\pi}{2} \mapsto \frac{2\alpha}{1+\alpha^2}, \quad \frac{\pi}{2} + i\zeta \mapsto 1, \quad -\frac{\pi}{2} + i\zeta \mapsto -1.$$

Since it holds for the Jacobi elliptic function sn that

$$-K \mapsto -1, \quad K \mapsto 1, \quad K + iK' \mapsto \frac{1}{k}, \quad -K + iK' \mapsto -\frac{1}{k},$$

it follows by comparison of formulas with $\pi K' = 2\zeta K$, i.e., $q = e^{-2\zeta}$ and $k(q) = \frac{2\alpha}{1+\alpha^2}$ that

$$t \circ w \circ \sin(z) = \frac{2\alpha}{1+\alpha^2} \cdot \operatorname{sn}\left(\frac{2K}{\pi}z, k(q)\right).$$

By Nehari [Neh75, p.293,(43)], it holds $\frac{2\sqrt{k(q^2)}}{1+k(q^2)} = k(q) = \frac{2\alpha}{1+\alpha^2}$, which gives $\alpha = \sqrt{k(q^2)}$ and finally

$$w(\sin(z)) = \sqrt{k(q^2)} \cdot \operatorname{sn}\left(\frac{2K}{\pi}z, q^2\right).$$

To obtain w , we take the inverse of \sin on z within the Jacobi elliptic sine function, resulting in $w(z) = \sqrt{k(q^2)} \cdot \operatorname{sn}\left(\frac{2K}{\pi}\sin^{-1}(z), q^2\right)$. In order to attain the same form of w as in Equation (A.1), we define $a := \cosh(\zeta)$ and $b := \sinh(\zeta)$. Next, we replace $q^2 = (e^{-2\zeta})^2$ with

$$\rho := q^2 = \left(\frac{a-b}{a+b}\right)^2$$

which is indeed the same, since $e^x = \cosh(x) + \sinh(x)$ and $\sinh(x) = \frac{e^x - e^{-x}}{2}$ as well as $\cosh(x) = \frac{e^x + e^{-x}}{2}$. Furthermore, it holds $a^2 - b^2 = \cosh(\zeta)^2 - \sinh(\zeta)^2 = 1$ and we obtain the same formula as in Equation (A.1):

$$w(z) = \sqrt{k(\rho)} \cdot \operatorname{sn}\left(\frac{2K}{\pi}\sin^{-1}\left(\frac{z}{\sqrt{a^2 - b^2}}\right), \rho\right) = \sqrt{k(\rho)} \cdot \operatorname{sn}\left(\frac{2K}{\pi}\sin^{-1}(z), \rho\right). \quad (\text{A.9})$$

A.2 Concrete calculations

The map w in Equation (A.9) is a precise conformal map from an ellipse E to the unit disk U . However, the function w is not immediately useful for calculating concrete coordinates due to involved integrals and infinite products. Therefore, we derive for a preimage point z_E how its image point w_U can be computed concretely.

For this, we start with Equation (A.9):

$$w: E \rightarrow U$$

$$z \mapsto \sqrt{k(\rho)} \cdot \operatorname{sn} \left(\frac{2K}{\pi} \sin^{-1} \left(\frac{z}{\sqrt{a^2 - b^2}} \right), \rho \right)$$

where

$$K = \int_0^{\frac{\pi}{2}} \frac{1}{1 - k^2 \sin^2 \varphi} d\varphi$$

$$k = \left(\frac{\theta_2(\tau)}{\theta_3(\tau)} \right)^2$$

$$\tau = \frac{2i}{\pi} \log \left(\frac{a+b}{a-b} \right) \quad \sim \quad \rho = e^{i\pi\tau} = \left(\frac{a-b}{a+b} \right)^2.$$

In Kober's work [Kob52, p.169], the theta functions θ_2 and θ_3 are defined by

$$\theta_2(\tau) = \vartheta_2(0, \tau)$$

$$\theta_3(\tau) = \vartheta_3(0, \tau)$$

which in turn depend on the functions ϑ_2 and ϑ_3 . On *MathWorld* [Wei23b], they are written in several forms:

$$\begin{aligned} \vartheta_2(z, \rho) &= \sum_{n=-\infty}^{\infty} \rho^{(n+\frac{1}{2})^2} e^{(2n+1)iz} = 2\sqrt[4]{\rho} \sum_{n=0}^{\infty} \rho^{n(n+1)} \cos((2n+1)z) = \\ &= 2G\sqrt[4]{\rho} \cos(z) \prod_{n=1}^{\infty} (1 + 2\rho^{2n} \cos(2z) + \rho^{4n}) \\ \vartheta_3(z, \rho) &= \sum_{n=-\infty}^{\infty} \rho^{n^2} e^{2niz} = 1 + 2 \sum_{n=1}^{\infty} \rho^{n^2} \cos(2nz) = \\ &= G \prod_{n=1}^{\infty} (1 + 2\rho^{2n-1} \cos(2z) + \rho^{4n-2}) \end{aligned} \tag{A.10}$$

for $\rho = e^{i\pi\tau}$ and $G = \prod_{n=1}^{\infty} (1 - \rho^{2n})$. Since k is given by the fraction of ϑ_2 and ϑ_3 , we select

the product forms where a lot cancels out when they are divided:

$$\begin{aligned} k &= \left(\frac{\vartheta_2(0, \rho)}{\vartheta_3(0, \rho)} \right)^2 \stackrel{\cos(0)=1}{=} \frac{2\sqrt[4]{\rho} \prod_{n=1}^{\infty} (1 + 2\rho^{2n} + \rho^{4n})}{\prod_{n=1}^{\infty} (1 + 2\rho^{2n-1} + \rho^{4n-2})} \\ &= \left(2\sqrt[4]{\rho} \prod_{n=1}^{\infty} \left(\frac{1 + 2\rho^{2n} + \rho^{4n}}{1 + 2\rho^{2n-1} + \rho^{4n-2}} \right) \right)^2 = 4\sqrt{\rho} \prod_{n=1}^{\infty} \left(\frac{1 + \rho^{2n}}{1 + \rho^{2n-1}} \right)^4. \end{aligned}$$

This formula for k can be calculated explicitly for a given $\rho = \left(\frac{a-b}{a+b}\right)^2$ as in Equation (A.9). Note that we have derived the same formula for k also given by Nehari [Neh75, p.292, (41)].

Given k , we study a concrete method for calculating the Jacobi elliptic function sn by

$$\operatorname{sn}(u, k) = \frac{\vartheta_3 \vartheta_1(u \vartheta_3^{-2})}{\vartheta_2 \vartheta_4(u \vartheta_3^{-2})}$$

[Wei23a, (12)] with $\vartheta_2 = \vartheta_2(0, \rho)$ and $\vartheta_3 = \vartheta_3(0, \rho)$ from Equation (A.10) and $\vartheta_i(z) = \vartheta_i(z, \rho)$ for $i = 1, 4$ as on *MathWorld* [Wei23b, (90),(93)] given by

$$\begin{aligned} \vartheta_1(z, \rho) &= 2G\sqrt[4]{\rho} \cos(z) \prod_{n=1}^{\infty} (1 - 2\rho^{2n} \cos(2z) + \rho^{4n}) \\ \vartheta_4(z, \rho) &= G \prod_{n=1}^{\infty} (1 - 2\rho^{2n-1} \cos(2z) + \rho^{4n-2}). \end{aligned} \tag{A.11}$$

Notice that $\frac{\vartheta_3}{\vartheta_2}$ equals $\frac{1}{\sqrt{k}}$ since k was defined to be $k = \left(\frac{\vartheta_3}{\vartheta_2}\right)^2$. Hence, for the function $w(z) = \sqrt{k} \cdot \operatorname{sn}(\dots)$, the factors \sqrt{k} and $\frac{\vartheta_3}{\vartheta_2}$ cancel and it only remains $\operatorname{sn}(u, k) = \frac{\vartheta_1(u \vartheta_3^{-2})}{\vartheta_4(u \vartheta_3^{-2})}$. It holds that $u = \frac{2K}{\pi} \sin^{-1}(z)$ (see Equation (A.9)). With Equation (A.11), we obtain the following:

$$w(z) = \frac{\vartheta_1(u \vartheta_3^{-2})}{\vartheta_4(u \vartheta_3^{-2})} = 2\sqrt[4]{\rho} \sin(u \vartheta_3^{-2}) \prod_{n=1}^{\infty} \left(\frac{1 - 2\rho^{2n} \cos(2u \vartheta_3^{-2}) + \rho^{4n}}{1 - 2\rho^{2n-1} \cos(2u \vartheta_3^{-2}) + \rho^{4n-2}} \right).$$

This is simplified by $\vartheta_3^2 = \frac{2K}{\pi}$ [Wei23b, (123)] which results in

$$u \cdot \vartheta_3^{-2} = \frac{2K}{\pi} \sin^{-1}(z) \cdot \frac{\pi}{2K} = \sin^{-1}(z).$$

Together with $\cos(2 \sin^{-1}(z)) = 1 - 2z^2$ and $\sin(\sin^{-1}(z)) = z$, we get

$$w(z) = 2\sqrt[4]{\rho} \cdot z \prod_{n=1}^{\infty} \left(\frac{1 - 2\rho^{2n}(1 - 2z^2) + \rho^{4n}}{1 - 2\rho^{2n-1}(1 - 2z^2) + \rho^{4n-2}} \right)$$

Appendix A. Conformal Map of an Ellipse to an Infinite Strip

with $\rho = \left(\frac{a-b}{a+b}\right)^2$. Since $a > b > 0$, it holds that $0 < \rho < 1$ and the powers of ρ within the infinite product ρ^{2n} , ρ^{4n} , ρ^{2n-1} and ρ^{4n-2} all converge to zero if $n \rightarrow \infty$. Hence, the sequence consisting of the factors of the product converges to 1. This is very convenient for the concrete computations of $w(z)$, since one can stop multiplying for $N \gg 1$ as soon as the factors of the infinite product are as close to 1 as machine precision.

To visualize these calculations, we again compute the map of an ellipse to the unit circle and use reverse pixel lookup to display the results. For each pixel located inside the ellipse, we calculate $w(z_e) = w_{e'}$ and obtain the color $c_{e'} = (r, g, b)$ at this pixel, which is displayed at pixel z_e .

Since our initial focus was on mapping the ellipse to an infinite strip, we use the resulting checkerboard image inside U from the above computations of the map m from the unit disk U to the infinite strip S (see Figure A.2). The final pattern of the concatenated maps w and m from the ellipse E to the infinite strip S within the ellipse can be seen in Figure A.5.

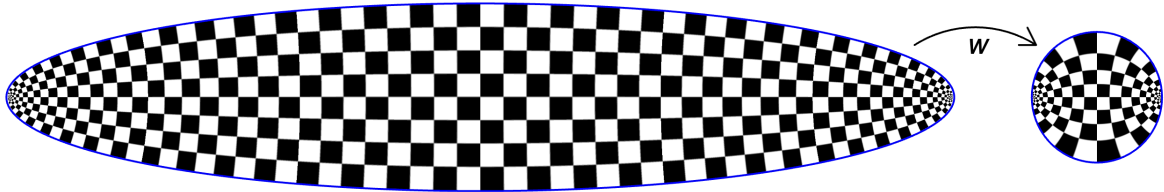


Figure A.5: Conformal map of an ellipse to the unit disk.

B | Logarithmic Spirals

Logarithmic spirals are spirals with equal angle between the tangent at any point on the spiral and the line connecting the point to the spiral's center. The spiral's radius grows by a constant factor with each winding, while the length from the center to any point on the spiral is finite. Descartes first studied logarithmic spirals in 1638, and Jakob Bernoulli was also fascinated by their properties [Rut18, p.71]. The equation for a basic logarithmic spiral $f: \mathbb{R} \rightarrow \mathbb{R}^2$ is given for constants $a, k \in \mathbb{R} \setminus \{0\}$ by

$$f(\varphi) := a \cdot e^{(k+i)\varphi} = ae^{k\varphi} \cdot \begin{pmatrix} \cos(\varphi) \\ \sin(\varphi) \end{pmatrix}. \quad (\text{B.1})$$

The spiral's center is given by $P := f(-\text{sign}(k) \cdot \infty)$ and referred to as the spiral's *pole*. In the basic definition of a logarithmic spiral in Equation (B.1), P is located at the origin $(0, 0)$ of the plane, but this may change later when the spiral is translated. The variable a is the distance between the pole P and $f(0)$ with $f(0)$ located on the x -axis at $(a, 0)$.

Parameter k determines the direction and the intensity of the spiral's winding around P . When k is positive, the spiral starts at $f(0)$ and moves towards P for decreasing $\varphi < 0$ in clockwise direction. Conversely, when k is negative, the spiral moves from $f(0)$ towards P for increasing $\varphi > 0$ in counterclockwise direction. The smaller the absolute value of k is, the closer the spiral is to a circle with radius a and the closer is the winding. This is because of the factor $r(\varphi) := ae^{k\varphi}$, which is called the *radius* of the spiral. For a full winding of 2π , the radius increases by the constant factor $e^{k \cdot 2\pi}$ since $r(\varphi + 2\pi) = ae^{k(\varphi + 2\pi)} = e^{k \cdot 2\pi} \cdot r(\varphi)$. Figure B.1 shows some basic logarithmic spirals f for different values of k .

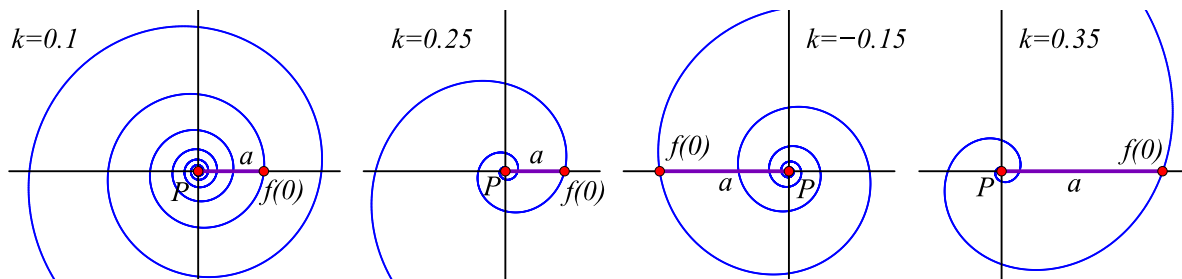


Figure B.1: Basic logarithmic spirals.

Parameter k also controls the constant angle between a tangent to the spiral in an arbitrary point $f(\varphi)$ and the line connecting $f(\varphi)$ and the pole. This angle α is given by $\alpha = \frac{\pi}{2} + \tan^{-1}(k)$, and it measures the counterclockwise angle between the tangent

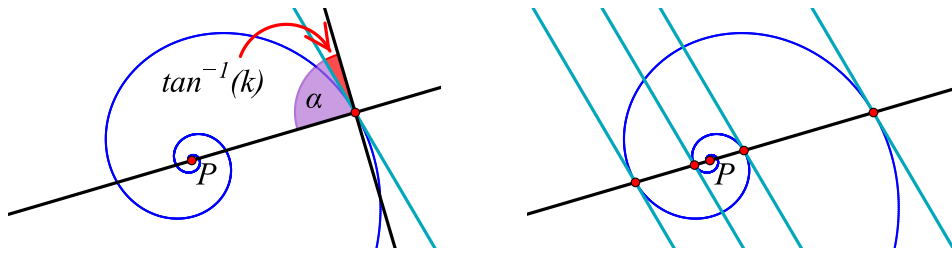


Figure B.2: Tangents including the same angle α with a line through the pole.

and the line (see Figure B.2 on the left). Since the angle α is independent from φ , the tangents at points $f(\varphi + j \cdot \pi)$ for $j \in \mathbb{Z}$ are parallel (see Figure B.2 on the right).

So far, we have only examined a logarithmic spiral in its basic form, which is neither translated nor rotated, with $P = (0, 0)$ and $f(0) = (a, 0)$. Now, we generalize this definition to include a translation to the point $A \in \mathbb{R}^2$ and a rotation around the origin $(0, 0)$ by the angle $\vartheta \in [-\pi, \pi)$, which is represented by a matrix $M \in \mathbb{R}^{2 \times 2}$:

$$\ell(\varphi) := ae^{k\varphi} \cdot \underbrace{\begin{pmatrix} \cos(\vartheta) & -\sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{pmatrix}}_{:=M} \begin{pmatrix} \cos(\varphi) \\ \sin(\varphi) \end{pmatrix} + A = ae^{k\varphi} \cdot \begin{pmatrix} \cos(\varphi + \vartheta) \\ \sin(\varphi + \vartheta) \end{pmatrix} + A. \quad (\text{B.2})$$

In comparison to the basic definition in Equation (B.1), the pole of the spiral is now translated to $P = A$. Additionally, in the initial equation, the point $f(0)$ of the spiral is situated on the x -axis at $(a, 0)$. For the rotated spiral, $\ell(0) - A$ is located on a circle around the origin with a radius of a at an angle of ϑ . The position of $\ell(0) - A$ on a circle justifies the restriction of the angle of rotation to the interval $[-\pi, \pi)$ caused by the periodicity of the cosine and sine functions. The property that all tangents enclose an identical angle α with a line passing through P remains unaltered by rotation and translation.

In addition to the tangentiality between a line and a logarithmic spiral, we are interested in the properties of logarithmic spirals that are tangent to each other. In Section B.1, we will explore various possibilities for placing two spirals so that they are tangent to each other. We will provide formulae for the explicit computation of a spiral ℓ_2 that is tangent to a given spiral ℓ_1 .

Building upon this knowledge, we will present a CPU-based digital pen simulation in Section B.2, which draws tangent logarithmic spirals along a user-drawn path.

Parts of the following are found in our paper [PRG21]. The complete code for the HTML applet of the digital spiral pen, named *SpiralPen_LP.html*, is available in [Pol23b].

B.1 Tangential logarithmic spirals

In general, logarithmic spirals are infinite curves defined by $\ell(\varphi)$ for $\varphi \in (-\infty, \infty)$ (see Equation (B.2)). To analyze tangency between spirals with possibly no further intersections apart from the point of contact, we limit the spirals to finite curves. This means,

we examine spirals between their starting point $\ell(0)$ and their pole P , i.e., we restrict the input parameters to $\varphi \in [0, \infty)$ if $k < 0$ and to $\varphi \in (-\infty, 0]$ if $k > 0$.

In the following, we investigate tangency between spirals ℓ_1 and ℓ_2 , which are defined by either the same parameter $k = k_1 = k_2$ or parameters $k_1 = -k_2$ that have opposite signs. We differentiate between the occurrence of one or more points of tangency and give formulas for the definition of ℓ_2 for the cases where, in addition to ℓ_1 , the point of tangency is given or the pole P_2 of the second spiral is known.

Spirals with equal parameters $k_1 = k_2$

Given a spiral ℓ_1 with parameter k_1 and pole P_1 , we search for a second spiral ℓ_2 with parameter $k_2 = k_1 := k$ and pole P_2 that is tangent to ℓ_1 . Initially, we assume the point of tangency to be given by $T_1 = \ell_1(\varphi_1)$ for $\varphi_1 \in (0, -\text{sign}(k) \cdot 2\pi)$, i.e., the spirals touch on the outermost winding of ℓ_1 . Without loss of generality, we define ℓ_2 so that $\ell_2(0) = \ell_1(\varphi_1)$.

By the definition of tangency, two tangent objects have the same tangent at the point of contact. Hence, the tangent t at $\ell_1(\varphi_1)$ and $\ell_2(0)$ must be the same. We know that t and the line between $\ell_1(\varphi_1)$ and P_1 enclose the angle $\alpha = \frac{\pi}{2} + \tan^{-1}(k)$ in counterclockwise direction. Since ℓ_2 is defined with the same parameter $k = k_2 = k_1$, t encloses the same angle α with the line between $\ell_2(0) = \ell_1(\varphi_1)$ and P_2 . Consequently, the lines connecting $\ell_1(\varphi_1) = \ell_2(0)$ to P_1 and P_2 have to be the same and P_2 lies on the line joining $\ell_1(\varphi_1)$ and P_1 (see Figure B.3).

Proposition B.1.1 *Two logarithmic spirals ℓ_1 and ℓ_2 with identical parameter k are tangent in a common point T_1 if and only if the poles P_1 and P_2 of the spirals and the point of contact T_1 are collinear.*

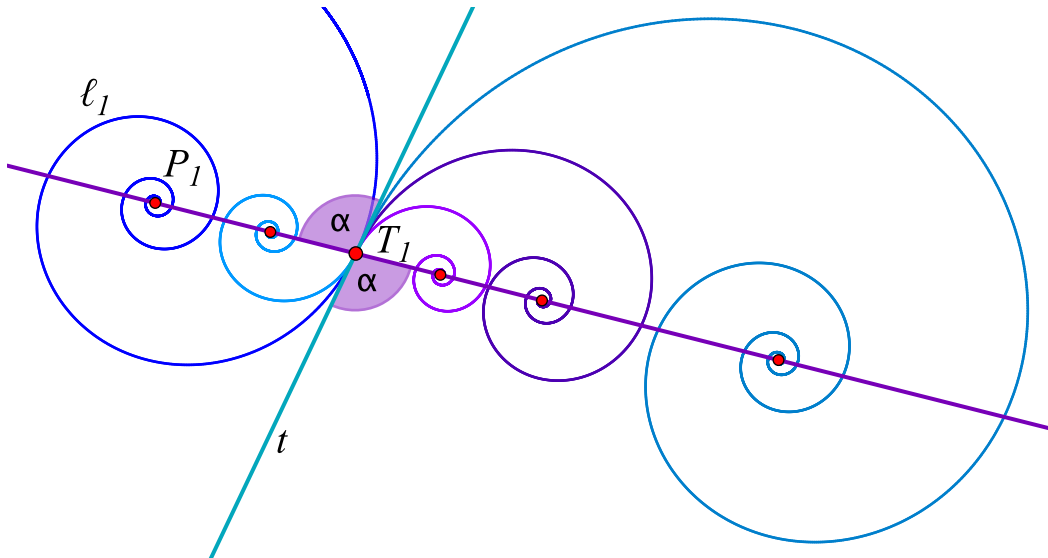


Figure B.3: The poles and the tangent point of the spirals are collinear.

Since ℓ_2 begins at $T_1 = \ell_1(\varphi_1) = \ell_2(0)$ and has parameter $k = k_1 = k_2$, the only piece of missing information needed to uniquely define the equation of ℓ_2 given by

$$\ell_2(\varphi) = a_2 \cdot e^{k\varphi} \cdot \begin{pmatrix} \cos(\varphi + \vartheta_2) \\ \sin(\varphi + \vartheta_2) \end{pmatrix} + P_2$$

is the position of P_2 . The pole P_2 can be freely selected on the line through P_1 and T_1 . Then, the parameter a_2 is given by $a_2 := \text{dist}(P_2, T_1) = |P_2 - T_1|$ and ϑ_2 is the angle that the vector $P_2 - T_1$ encloses with the x -axis. In CindyJS [RGK23b], the function `arctan2` can be used to calculate this angle $\vartheta_2 := \text{arctan2}(P_2 - T_1)$.

We observe in Figure B.3 that the pole P_2 of spiral ℓ_2 may be situated on either the opposite or same side of the tangent t as the pole P_1 of spiral ℓ_1 . If we define P_2 to be located on the opposite side of T_1 than P_1 , the two spirals are symmetric with respect to T_1 if $a_1 = a_2$, i.e., if $P_2 = T_1 + (T_1 - P_1)$. If we additionally define T_1 as the starting point of both spirals, i.e., $T_1 = \ell_1(0) = \ell_2(0)$, the resulting symmetric double spiral looks like one of the examples in Figure B.4.

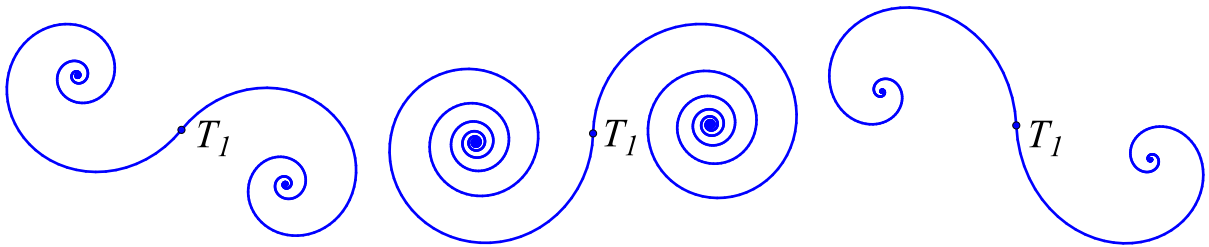


Figure B.4: Symmetric double spirals for different parameters k .

If we define P_2 to be on the same side of T_1 as P_1 , we can determine where P_2 should be located to allow for the two spirals ℓ_1 and ℓ_2 to have a second point of contact as in Figure B.5 on the left. We call this second point of contact T_2 . We know from Proposition B.1.1 that T_2 must be on the same line as T_1 , P_1 and P_2 since the angle α is the same throughout ℓ_1 and ℓ_2 . Hence, T_2 is given by

$$T_2 = \ell_2(-\text{sign}(k) \cdot \pi) = \ell_1(\varphi_1 - \text{sign}(k) \cdot 2\pi).$$

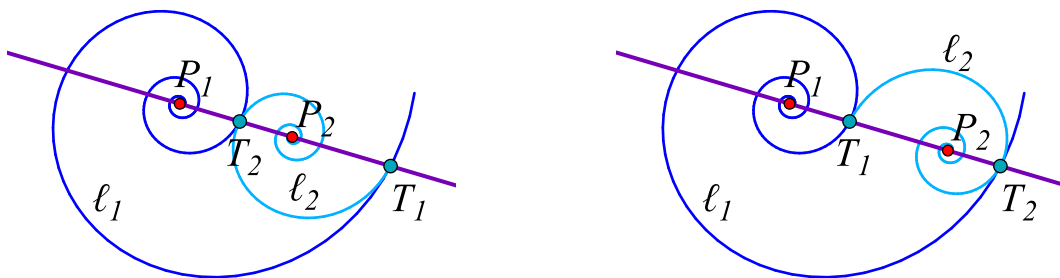


Figure B.5: Two spirals ℓ_1 and ℓ_2 with two points of contact T_1 and T_2 .

When T_1 and T_2 are given, the position of P_2 is fixed.

Proposition B.1.2 *Let $\ell(\varphi)$ be a logarithmic spiral with known parameter $k \neq 0$. Let two points on the spiral be $T_1 = \ell(0)$ and $T_2 = \ell(-\text{sign}(k) \cdot \pi)$. Then the pole P of spiral ℓ is located at*

$$P = T_1 + \frac{e^{|k| \cdot \pi}}{(e^{|k| \cdot \pi} + 1)} \cdot (T_2 - T_1).$$

Proof: Since $T_1 = \ell(0)$, its distance to the pole P defines the parameter $a := \text{dist}(T_1, P)$. The distance between the pole and $T_2 = \ell(-\text{sign}(k) \cdot \pi)$ equals $a \cdot e^{-|k|\cdot\pi}$ by the definition of ℓ in Equation (B.2). Since the pole P and the points T_1 and T_2 are collinear, it holds that $\text{dist}(T_1, P) + \text{dist}(T_2, P) = \text{dist}(T_1, T_2)$. Hence, it holds

$$\text{dist}(T_1, P) = e^{|k|\cdot\pi} \cdot \text{dist}(T_2, P) \quad \text{and} \quad \text{dist}(T_1, T_2) = (e^{|k|\cdot\pi} + 1) \cdot \text{dist}(T_2, P).$$

Inserting the equation for $\text{dist}(T_1, T_2)$ into the equation for $\text{dist}(T_1, P)$ results in

$$\text{dist}(T_1, P) = \frac{e^{|k|\cdot\pi} \cdot \text{dist}(T_1, T_2)}{(e^{|k|\cdot\pi} + 1)}$$

which concludes the proof. \square

If the roles of T_1 and T_2 are interchanged, we get a spiral ℓ_2 tangent to ℓ_1 as in Figure B.5 on the right. The formula for the pole from Proposition B.1.2 remains unchanged.

Another possibility to create tangent logarithmic spirals is to define

$$T_1 = \ell_2(0) = \ell_1(-\text{sign}(k) \cdot 2\pi) \quad \text{and} \quad T_2 = \ell_1(0) = \ell_2(-\text{sign}(k) \cdot 2\pi).$$

This leads to intertwined spirals like the ones shown in Figure B.6. If ℓ_1 is given, the position of P_2 is calculated by

$$P_2 = T_1 + \frac{1}{(1 - e^{-|k|2\pi})} \cdot (T_2 - T_1).$$

The proof of this follows the same argumentation as the proof of Proposition B.1.2 for $a_2 := \text{dist}(T_1, P_2)$ and $\text{dist}(T_1, T_2) + \text{dist}(T_2, P_2) = \text{dist}(T_1, P_2)$.

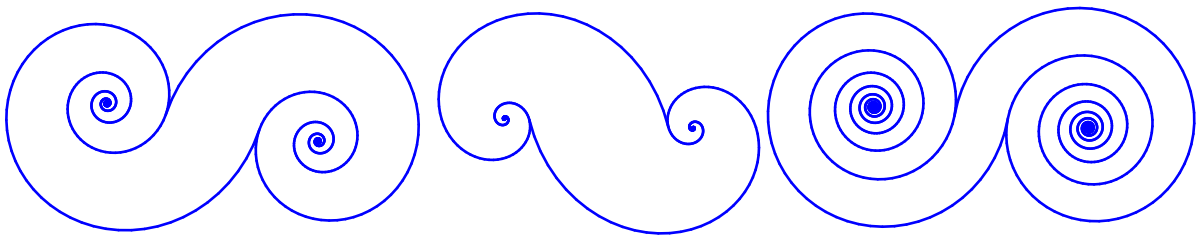


Figure B.6: Intertwined tangent spirals for different parameters k .

Another related problem is to find the tangent spiral ℓ_2 when the spiral ℓ_1 as well as the pole P_2 of the second spiral are given. So far, we calculated P_2 from a given point of contact. Now we have to calculate the common point T of ℓ_1 and ℓ_2 . Again, without loss of generality, we assume that $T = \ell_2(0)$. Hence, once T is given, we can define ℓ_2 with $a := \text{dist}(T, P_2)$ and $\vartheta := \arctan2(P_2 - T)$.

We know from Proposition B.1.1 that P_1 , P_2 and T must be collinear. So we need to find the unique angle $\varphi_1 \in (0, -\text{sign}(k) \cdot 2\pi)$ for which it holds that the line through P_1 and P_2 intersects ℓ_1 in $T = \ell_1(\varphi_1)$.

The angle φ_1 is determined by the vectors $v = P_2 - P_1$ and $w = \ell_1(0) - P_1$. Typically, the formula $\sphericalangle = \arccos\left(\frac{v^T w}{|v| \cdot |w|}\right)$ is used to calculate the angle between two vectors.

This formula always produces a positive angle smaller than π between the two vectors, i.e., $\sphericalangle \in (0, \pi)$. This, however, is not applicable here as we are looking for φ_1 in the interval $(0, -\text{sign}(k) \cdot 2\pi)$. The correct sign of φ_1 can be added when the absolute value of φ_1 is known. To obtain the absolute value between zero and 2π , we calculate the scalar product of w with $v^\perp := (-v_2, v_1)^T$. If $w^T v^\perp > 0$, the counterclockwise angle between v and w is smaller than π . If $w^T v^\perp < 0$, this counterclockwise angle is greater than π . In Figure B.7 on the left, blue color indicates regions where $w^T v^\perp > 0$, while purple color indicates regions where $w^T v^\perp < 0$. With this distinction, we have to consider the sign of the parameter k . If $k > 0$, ℓ_1 moves clockwise towards P_1 . If $k < 0$, the spiral ℓ_1 moves counterclockwise towards P_1 . This means, that the absolute value of φ_1 equals \sphericalangle when $k > 0$ and $w^T v^\perp > 0$. Alternatively, it equals $2\pi - \sphericalangle$ for $k > 0$ and $w^T v^\perp < 0$. For $k < 0$, it is the other way round. The middle and right images in Figure B.7 illustrate how the orientation of the spiral influences the absolute value of φ_1 . After calculating the absolute value, φ_1 is finally given by $\varphi_1 = -\text{sign}(k) \cdot |\varphi_1|$. Algorithm 11 presents the pseudocode for calculating φ_1 from P_1 , P_2 and $\ell_1(0)$.

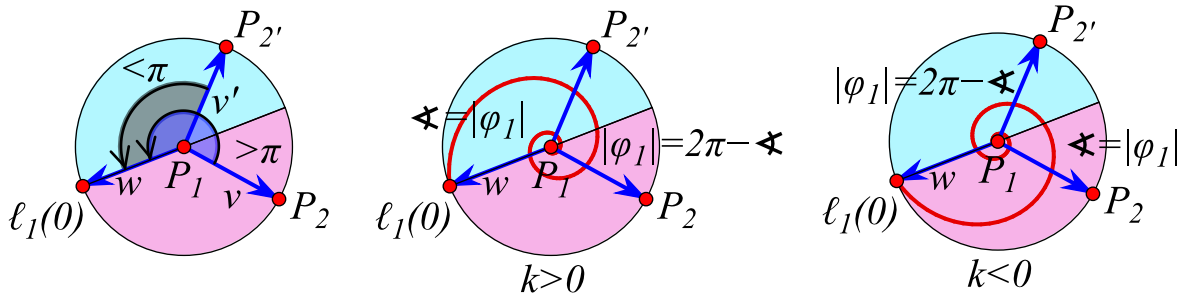


Figure B.7: Calculating the angle between $P_2 - P_1$ and $\ell_1(0) - P_1$.

Algorithm 11 Calculating φ_1 from P_1 , P_2 and $\ell_1(0)$.

```

 $v \leftarrow P_2 - P_1; \quad w \leftarrow \ell_1(0) - P_1$ 
 $v^\perp \leftarrow (-v_2, v_1)^T$ 
 $\sphericalangle \leftarrow \arccos\left(\frac{v^T w}{|v| \cdot |w|}\right)$ 
if  $w^T v^\perp \geq 0$  then
    if  $k > 0$  then  $|\varphi_1| \leftarrow \sphericalangle$ 
    else  $|\varphi_1| \leftarrow 2\pi - \sphericalangle$ 
    end if
else if  $w^T v^\perp < 0$ 
    if  $k > 0$  then  $|\varphi_1| \leftarrow 2\pi - \sphericalangle$ 
    else  $|\varphi_1| \leftarrow \sphericalangle$ 
    end if
end if
 $\varphi_1 \leftarrow -\text{sign}(k) \cdot |\varphi_1|$ 
    
```

The point of tangency T is finally given as $T = \ell_1(\varphi_1)$ and the tangent spiral ℓ_2 is uniquely defined for $\ell_2(0) = T$ and pole P_2 . The resulting pair of spirals looks like one of the pairs shown in the previous figures depending on where P_2 is located.

Spirals with opposite parameters $k_1 = -k_2$

Instead of two spirals ℓ_1 and ℓ_2 with the same parameter $k = k_1 = k_2$, we will now examine two spirals with parameters k_1 and k_2 that have opposite signs, that is, $k_1 = -k_2$. This changes the behavior of tangency at the common point T . For the counterclockwise angles α_1 and α_2 formed between the tangent t and the lines connecting T and the poles P_1 and P_2 , respectively, it holds that

$$\alpha_1 + \alpha_2 = \left(\frac{\pi}{2} - \tan^{-1}(k_1)\right) + \left(\frac{\pi}{2} - \tan^{-1}(k_2)\right) = \left(\frac{\pi}{2} - \tan^{-1}(k_1)\right) + \left(\frac{\pi}{2} + \tan^{-1}(k_1)\right) = \pi.$$

Let m_1 be line TP_1 and let m_2 be line TP_2 . The function $\text{join}([point_1], [point_2])$ computes the line between two given points. Hence, it holds $m_1 = \text{join}(T, P_1)$ and $m_2 = \text{join}(T, P_2)$. It follows from $\alpha_1 + \alpha_2 = \pi$ that m_1 and m_2 must be reflections of each other with respect to t . This reflection guarantees that ℓ_1 and ℓ_2 have a common tangent t in T . Figure B.8 shows an example.

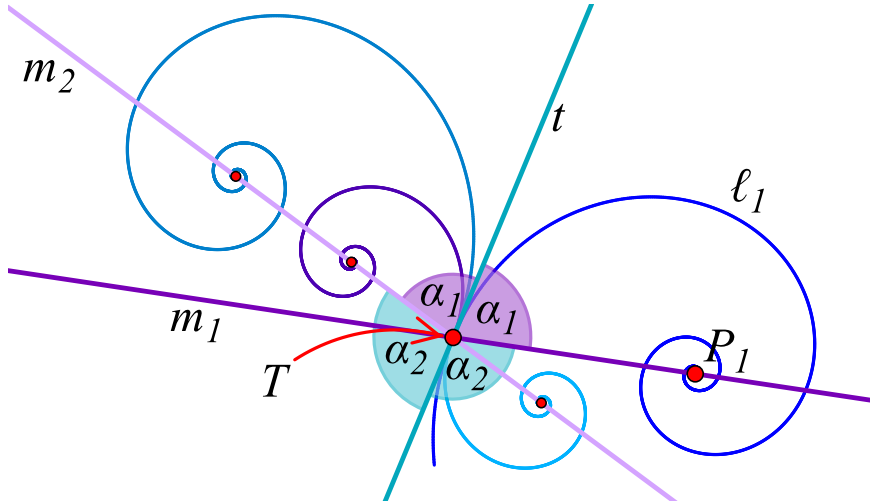


Figure B.8: The lines m_1 and m_2 are reflections with respect to the tangent t .

Proposition B.1.3 *Two logarithmic spirals ℓ_1 and ℓ_2 with parameters $k_1 = -k_2$ share the same tangent t in the common point T if and only if the poles P_1 and P_2 lie on lines $m_1 = \text{join}(T, P_1)$ and $m_2 = \text{join}(T, P_2)$, respectively, which are reflections of each other with respect to t .*

If the common point T on ℓ_1 is given, we can define without loss of generality that $T = \ell_2(0)$ and choose P_2 on m_2 freely. This uniquely defines $\ell_2(\varphi)$ with $a := \text{dist}(T, P_2)$ and $k_2 = -k_1$. The rotation angle is given by $\vartheta_2 := \arctan2(P_2 - T)$ as for the case when $k_1 = k_2$.

If P_2 is given instead of T , we have not yet found an easy formula for T on ℓ_1 such that $m_1 := \text{join}(T, P_1)$ and $m_2 := \text{join}(T, P_2)$ are reflections of each other. Therefore, we must efficiently search for the correct point T on ℓ_1 . For this, we observe that the point T always lies between $\ell_1(0)$ and $\ell_1(-\text{sign}(k) \cdot \theta)$, where θ is the angle between the vectors $(\ell_1(0) - P_1)$ and $(P_2 - P_1)$ (see Figure B.9). Hence, starting at θ , we take small

steps with a size of $\epsilon > 0$ and examine whether $S = \ell_1(-\text{sign}(k) \cdot (\theta - \epsilon))$ satisfies the condition that $\text{join}(S, P_1)$ and $\text{join}(S, P_2)$ are reflections with respect to tangent t_S at S . It is also possible to do a search of nested intervals.

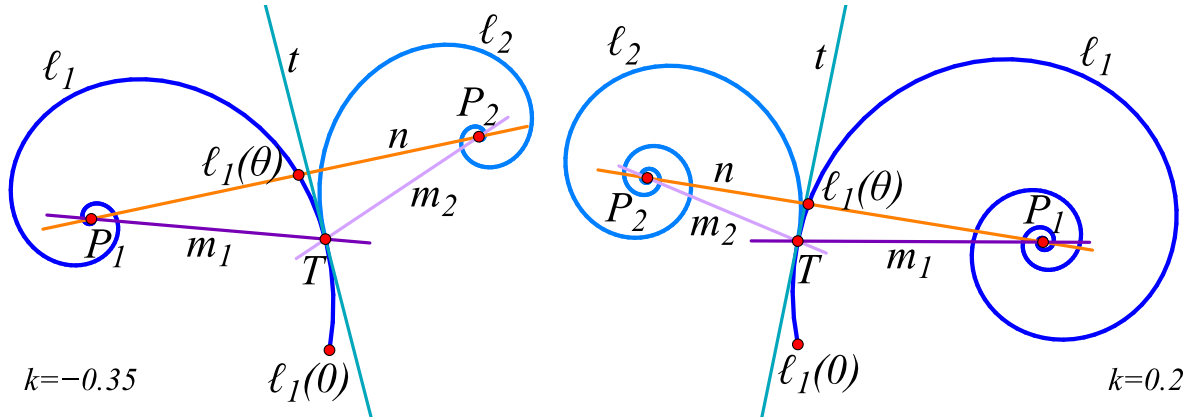


Figure B.9: T lies between $\ell_1(0)$ and $\ell_1(\theta)$ on $\text{join}(P_1, P_2)$.

Combining tangential spirals with both same and opposite parameters of k is a suitable method for designing floral structures like those presented in Section 6.2.1.

B.2 A CPU based spiral pen

With the knowledge from the previous section, we implemented a CPU-based pressure-sensitive pen simulation. The fundamental structure of circles along a user-drawn path is identical to that of the stroke model described in the main text. Hence, the registered discrete data points P_{t_d} along the user-drawn path are given as input. The radius values at these registered data points are calculated using Equation (5.1), based on the pressure and tilt information of the digital pen and the slider settings for the constant stroke width, as well as the percentage of pressure and tilt influence. The points P_{t_d} are used as input for an interpolation of a curve γ on which equidistant points are calculated as in Algorithm 1. In contrast to the cubic spline interpolation of γ in the algorithm for the stroke model presented in the main text, we use only linear interpolation here (see Section 4.1.1). Linear interpolation is also used to calculate the radius values corresponding to the circles centered at the computed equidistant curve points. This results in a discrete family of circles along the user-drawn path (see Figure B.10 in the background).

The pressure-sensitive digital pen model presented in the main text was enriched by a conformally mapped ornamental design. In contrast, we now want to create stamps from logarithmic spirals placed tangentially next to each other in the stroke given by the family of circles as shown in Figure B.10. These spirals adapt their position and size according to the circles' radii as the stroke is drawn.

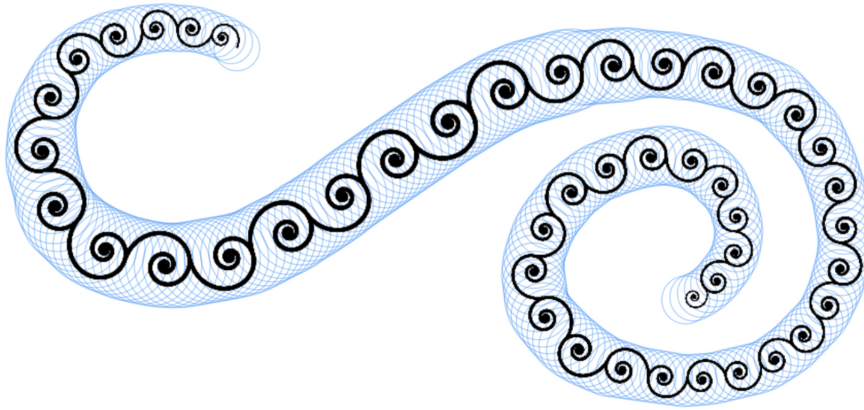


Figure B.10: Tangent spirals placed inside a stroke defined by a family of circles.

To select the appearance of the spiral stamps, a user interface next to the drawing surface is provided. The parameter $k \in [-0.5, 0.5]$ is chosen with a slider. For the appearance of the printed spirals, the user can choose from several options.

- 1) One option is to alternately print a single logarithmic spiral with a chosen parameter k and its negative $-k$ along the stroke, as illustrated in Figure B.10. In addition to the main spiral, it is possible to add an inner spiral with the same parameter k that touches twice, as in Figure B.5 on the right. A slider determines the position of the inner spiral relative to its starting point on the outermost winding of the main spiral. Also, a spiral with the opposite parameter $-k$ can be added, touching the main spiral from the outside, as in Figure B.9. Another slider controls its position on the outermost winding of the main spiral. See Figure B.11 on the left for the user interface of an exemplary selection of a single spiral stamp.
- 2) Another possibility is to choose a combination of two symmetrical spirals with the same parameter k , as shown in Figure B.4. There are two additional options, one of which must be selected: either the same pair of spirals is placed tangentially along the stroke, or the pair is alternately placed once with parameter k and once with parameter $-k$. See Figure B.11 in the middle for an example of the user interface for this option.
- 3) The third alternative is to create a stamp of two intertwined spirals, as shown in Figure B.6. Similar to option 2), a pair of intertwined spirals can be placed repeatedly along the stroke, or alternated with a pair of intertwined spirals of opposite parameter $-k$. See Figure B.11 on the right for an instance of the user interface for this case.

Exemplary strokes for all three cases drawn with a pressure-sensitive digital pen are shown in Figure B.12.

For case 1), we included a pressure sensitivity condition when inner and/or outer spirals are chosen. If the distance between the pole and the starting point of the supplementary inner or outer spirals is less than a predetermined value, they will not be displayed despite being selected in the user interface. This feature is related to the Chapter 6 where adaptive design concepts that depend on pressure sensitivity are presented.

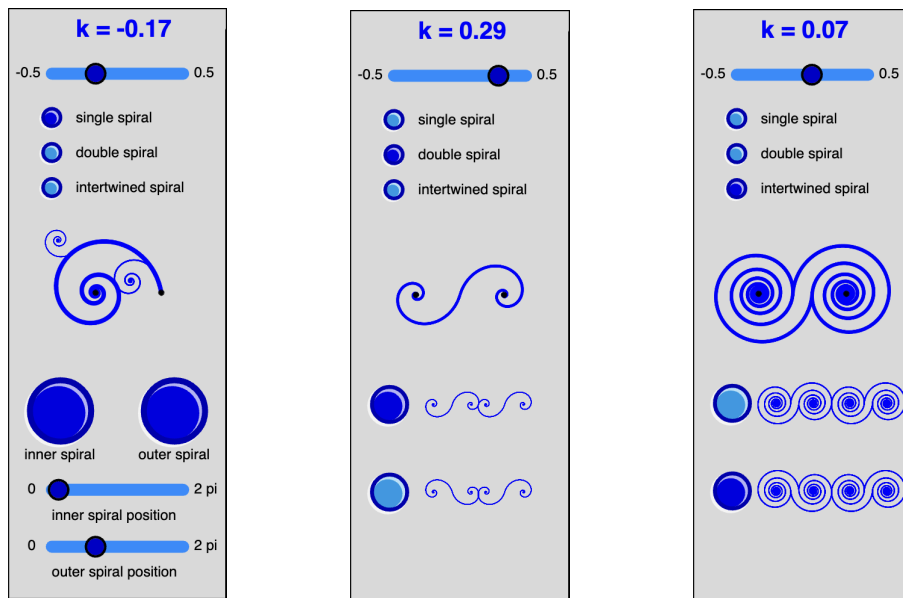


Figure B.11: User interface to select different types of tangential spirals.

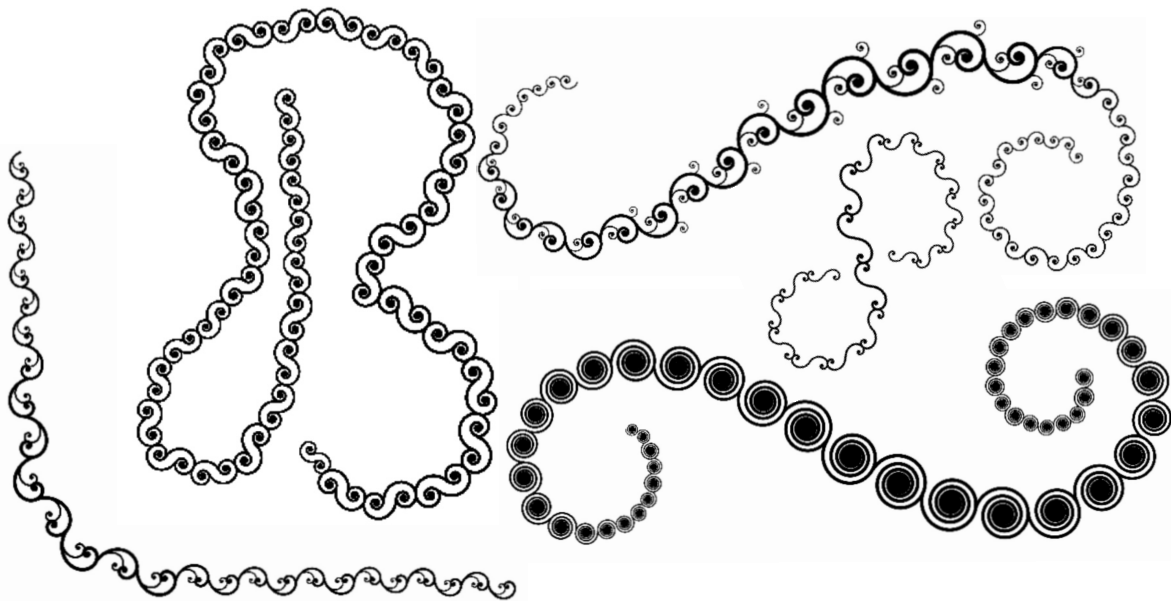


Figure B.12: Strokes with different types of tangential spirals.

It is important to note that the individual spirals are placed with a stamp, not deformed to fit the stroke's appearance. Therefore, the outer spiral in case 1) may exit the region of the stroke defined by the family of circles. The symmetrical or intertwined spiral pairs in cases 2) and 3) may also leave the stroke, especially in stroke regions with narrow turns. Consequently, the map of the tangential spirals onto the stroke is not conformal. Nonetheless, the outcomes remain aesthetically pleasing.

Implementation details of the spiral pen

In this section, we provide additional details on the implementation of the spiral pen in CindyJS [RGK23b]. The required information to print the spirals in the stroke region is derived from the selections in the user interface (see Figure B.11), from the linearly interpolated input data and from the selected equidistant curve points with corresponding radii. This information is stored in a list called `spiralList` which contains an entry for each individual spiral. The last entry `spiralListn` is updated when the corresponding spiral or pair of spirals in the stroke changes its position and size due to the growing stroke. Once this (pair of) spiral(s) reaches its final position and size, the entry remains unchanged, and a new entry `spiralListn+1` is added to `spiralList` for a new upcoming (pair of) spiral(s).

To draw a spiral, it is necessary to know the location of its pole P and its starting point $\ell(0)$. Together, these define the parameters $a := \text{dist}(P, \ell(0))$ and k , and the rotation matrix M (see the definition of a logarithmic spiral in Equation (B.2)). For the calculations it is also convenient to know the common point of two tangential spirals.

We assume that we are in case 1) of single spirals without additional inner or outer spirals. We further assume that a spiral ℓ_n has reached its final position and a new entry `spiralListn+1` for the new spiral ℓ_{n+1} was added to `spiralList`. This entry will contain four pieces of information that are important for drawing the new spiral: the pole of the new spiral P_{n+1} , the spiral's starting point $\ell_{n+1}(0)$ which is also the point of contact with the spiral ℓ_n , the parameter $k_{n+1} = -k_n$, and the rotation matrix M_{n+1} .

When a new equidistant point $\gamma(t_i)$ is placed on the interpolated user-drawn curve, the pole P_n and starting point $\ell_n(0)$ of the prior spiral are read from `spiralListn`. The appropriate angle φ_n between $v = \ell_n(0) - P_n$ and $w = \gamma(t_i) - P_n$ is then calculated (see Algorithm 11). This angle φ_n is used to determine whether $\gamma(t_i)$ is located outside the spiral ℓ_n . If this is true, a new spiral ℓ_{n+1} is defined. For this, we compute $r_n = a_n \cdot e^{-|k_n|\varphi_n}$. If $r_n < \text{dist}(\gamma(t_i), P_n)$, the new point $\gamma(t_i)$ is suitable to define the new spiral ℓ_{n+1} . Since $k_{n+1} = -k_n$, the next step is to search for point $\ell_{n+1}(0)$ on ℓ_n so that ℓ_{n+1} is tangential to ℓ_n at that point. As stated in the end of Section B.1, the search begins at $\ell_n(\varphi_n)$ and continues until the correct point is found. This point $\ell_{n+1}(0)$ together with $\gamma(t_i) = P_{n+1}$, the parameter k_{n+1} and the rotation matrix M_{n+1} deduced from $\ell_{n+1}(0)$ and P_{n+1} are stored in `spiralListn+1`. When the first spiral is defined, i.e., when the entry `spiralList1` is filled with information, the curve point $\gamma(t_0)$ is automatically taken as the start point of this first spiral. The described procedure is visualized in Figure B.13.

The procedure is repeated each time a new point $\gamma(t_i)$ is defined. A spiral ℓ_{n+1} reaches its final position and size when $|\text{dist}(\ell_{n+1}(0), P_{n+1}) - r_i| < \epsilon$ for $\epsilon > 0$ and r_i being the radius of the current circle around $\gamma(t_i)$. If $\text{dist}(\ell_{n+1}(0), \gamma(t_i)) > r_i + \epsilon$, the pole P_{n+1} is searched on the line segment between the currently stored P_{n+1} and $\gamma(t_i)$. Once ℓ_{n+1} is finalized, a new entry is added to `spiralList`.

If the user selects tangential inner and/or outer spirals in addition to the main spirals, these spirals are calculated separately for each main spiral ℓ_n according to the information given in the user interface. The necessary formulas for both inner and outer spirals, with the same and opposite parameters k compared to ℓ_n , respectively, were presented in the preceding Section B.1.

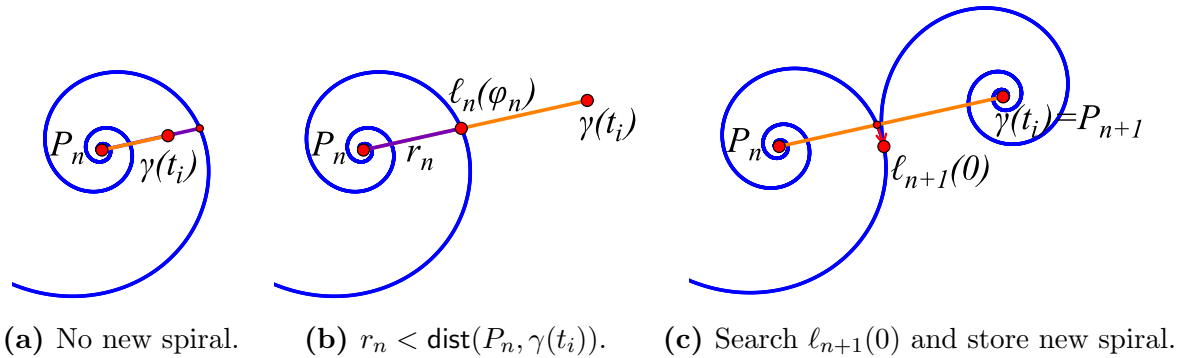


Figure B.13: Definition of a new spiral in the stroke.

If the user selects case 2) of a symmetric double spiral or case 3) of an intertwined pair of spirals, the procedure is essentially identical to that of case 1). However, each new entry in `spiralList` has to contain more pieces of information than in case 1) since every new structure consists of a pair of spirals. Each entry now stores two poles and two rotation matrices. There is still only one point of contact to the previous pair of spirals. The two spirals within one pair share the same parameter k . Storing the starting point of one spiral in a pair is sufficient because the other is either the same (case 2)) or can be computed from the poles, the given starting point, and the point of contact with the previous pair of spirals (case 3)).

Furthermore, depending on the selected option in the user interface, two consecutive pairs of spirals can have either the same or opposite parameters k . Hence, the point of contact of the new pair of spirals can be directly obtained by calculating the angle φ_n at the outermost intersection point of the spiral ℓ_n with the line passing through P_n and P_{n+1} if $k_1 = k_2$. Otherwise, it must be calculated on the second spiral of the previous pair between the intersection and the starting point of the second spiral of the previous pair.

Finally, the criterion for determining when a pair of spirals has reached its final position must be adjusted to the fact that the pair is considered as one unit and not as two separate spirals. In case 2), the point $\gamma(t_i)$ does not serve as the second pole of the spiral. Instead, both new poles are positioned between $\gamma(t_i)$ and the point of contact on the previous spiral. The starting point of both spirals is determined as the midpoint of $\gamma(t_i)$ and the touching point. The spirals reach their final size when the distance between the touching point and $\gamma(t_i)$ is approximately $3 \cdot r_i$, where r_i is the radius of the current circle around $\gamma(t_i)$. In case 3), the pole of the second spiral in the intertwined pair is defined as $\gamma(t_i)$. The distance between $\gamma(t_i)$ and the starting point of the second spiral in the new pair is calculated. If it is approximately equal to the current radius r_i , the spiral is fixed. Otherwise, it is altered in the subsequent iteration when a new curve point is calculated.

According to the selection in the user interface, a stamp is created for the respective spiral structure, i.e., a single spiral with or without inner and outer spirals, a double spiral with the same starting point, or a pair of intertwined spirals. This stamp consists of a texture, i.e., a pixel image, on which the respective spirals are drawn. Additionally, a second stamp is created that contains the same spirals as the first stamp, but with

parameters k of opposite sign. This is necessary for creating a stroke consisting of single spirals and for cases 2) and 3), when the pairs of spirals are intended to have alternating orientations. The position of the poles and starting points of the involved spirals is stored for both stamps and required for printing them onto the drawing surface at the correct location. A similarity transformation is computed for each spiral in the stroke, which maps the spirals in the stamp to the corresponding spirals in the stroke. Then, the spirals of the stamp are printed onto the drawing surface at the positions calculated by this transformation.

Note that printing a stamp texture onto the drawing surface is faster than using the CindyJS function `plot` for each spiral in the stroke.