AutoGraff:

towards a computational understanding of graffiti writing and related art forms.

Daniel Berio

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
Goldsmiths, University of London.

Department of Computing
Goldsmiths, University of London

January 10, 2021

I, Daniel Berio, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The aim of this thesis is to develop a system that generates letters and pictures with a style that is immediately recognizable as graffiti art or calligraphy. The proposed system can be used similarly to, and in tight integration with, conventional computer-aided geometric design tools and can be used to generate synthetic graffiti content for urban environments in games and in movies, and to guide robotic or fabrication systems that can materialise the output of the system with physical drawing media.

The thesis is divided into two main parts. The first part describes a set of *stroke primitives*, building blocks that can be combined to generate different designs that resemble graffiti or calligraphy. These primitives mimic the *process* typically used to design graffiti letters and exploit well known principles of motor control to model the way in which an artist moves when incrementally tracing stylised letterforms. The second part demonstrates how these stroke primitives can be automatically recovered from input geometry defined in vector form, such as the digitised traces of writing made by a user, or the glyph outlines in a font. This procedure converts the input geometry into a seed that can be transformed into a variety of calligraphic and graffiti stylisations, which depend on parametric variations of the strokes.

Acknowledgements

My fascination with graffiti art started with me being a kid that was trying to learn the C-programming language. I saw graffiti tags in the streets of my hometown, Florence, and I wandered what was this obscure code written on the walls of the city. Ironically, one of my first attempt at sketching graffiti letters, was writing the word “UNIX” on a schoolbook. Little did I know, that I would end up writing graffiti for years to come but also end up writing this massive document on the combined topic of graffiti and computing.

This course of events would have not been possible without the support and advice of my supervisor Frederic Fol Leymarie. I would like to thank him immensely for believing in this project, for his friendship, for his mentoring and for being engaged with the work until the very last minute. I would also like to thank Frederic for sharing with me his expertise, insight and opinions about shape, which have greatly contributed to this work and I hope are expressed to his satisfaction in this thesis.

The connections of the thesis to “skeletons” (in the sense of Harry Blum) goes back to my friend Alexander Bucksch, who I would like to thank for the good times back in Den Haag, for convincing me in the first place to pursue a PhD and for warning me, only when my proposal was accepted, about the potential downsides of this life choice. I would also like to thank Prashant (Alpha) Aparajeya for the good times and conversations during the start of my studies and for always being more reliable than Wolfram Alpha in answering my mathematical questions. I would also like to thank Alex Evans, for always being supportive during my PhD and for having in a way set course to this journey by enabling me to visit Guildford in the early 2000s.

A special thank-you also goes to Sylvain Calinon, who is not listed as an official supervisor, but most certainly helped me as such. A lot of the work in this thesis is also due to his contributions and advice, and I would also thank him for letting me visit the Idiap research institute in Switzerland, where I spent 6 months struggling with the challenges of controlling a humanoid robot and diving in a lot of the theoretical background underlying this thesis. In this regard, I would like to thank in particular Réjean Plamondon, who’s theories have also contributed to three chapters of this thesis. I would like to thank him firstly for believing in this project, and also for his support, advice and for introducing me to the Graphonomics

community. I would like to also thank Tamar Flash, for her kindness, the always stimulating conversations and for pointing me in interesting new directions, which I hope to explore more in depth in the future. My gratitude also goes to Rebecca Chamberlain and Luca Citi, who's additional supervision and ideas have brought the work presented here in interesting new directions. In particular I would like to thank Rebecca, together with Guido Orgs and Caitlin Mullin and collaborators for also greatly contributing to this project with their excellent empirical research.

My gratitude also goes to Paul Asente and Jose Echevarria at Adobe Research, for supporting this project in the last two years, and for the precious advice during the most challenging part of my studies. In particular I would like to thank Paul for also believing in this project, always challenging me with his questions and letting me visit Adobe in the first place, where another big part of the work in this thesis was also conceived. I would also like to thank Daichi Ito at Adobe Research for providing beautiful designs to test with our system and for presenting results in Japan and in Korea.I am also grateful to my examiners Craig Kaplan and Anthony Steed for taking their time to read this document in depth, for the stimulating discussion during the viva and for the useful and sometimes even hilarious comments.

A big thank-you goes also to the IGGI (Intelligent Games and Game Intelligence) programme for supporting this PhD and in particular to my Goldsmiths colleagues and friends Christian Guckelsberger, Memo Akten and Tom Cole, with a longing for our restaurant explorations (and feasts) during the first year of the PhD studies, as well as Henrik Siljebrat and Rob Homewood for the fun after-lab sessions at the pub. I would like to especially thank Memo, who contributed in a big part to one chapter of this thesis and with whom I shared sleepless nights trying to finish the work in time for a deadline. I want to also thank my friend Yaprak for making me smile during tough times, and also send a thank-you to a number of graffiti artists, some for contributing with photos or suggestions and some simply for being good friends during this time, as we say for graffiti I will “put them up” with their tags as BEES, SMART, KEIN, MORE, HULK, VIME, RELAX, EGS, CESAR, KRESO, GREY, TRIXTER, NEMA, ELK, DRAX, SIEGE, PETRO, YEP, RAKIE, YOM and probably more that I forgot to add here.

Finally, this thesis is dedicated to my mother (Ima) my brother (Yoni) and in loving memory of my father (Aba).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this thesis is to develop a set of parametric primitives and methods for the computer aided design of graffiti art. What do I mean by graffiti art? The term graffiti generally refers to any form of writing done on a surface without authorisation (Kimvall, 2007). In this study I will use this term specifically to refer to the art movement — also known as *Writing, Aerosol Art, Spray Can Art* – that revolves around various forms of stylisation and abstraction applied to the letters of a *tag*, a pseudonym for an artist who is often referred to as a (graffiti) *writer*. This art form emerged in the late 1960s when tags started to appear on the surfaces of the New York City subway (Kimvall, 2014; Ferri, 2016), and developed into a rich and complex art form that can be seen today on walls and surfaces around the globe. The methods presented in this thesis are not specifically aimed at producing real instances of graffiti art, but rather at facilitating the conception and generation of graffiti designs with computational techniques.

Graffiti art evolved from initially simple signatures to complex and colorful letter designs. These designs distill a number of graphic influences, ranging from typography and calligraphy to comics and science fiction just to name a few. The interplay of these influences, together with the mutual inspiration between different artists can be seen as a form of "Chinese whispers of style" [1] that has produced the highly distinctive aesthetic that characterises graffiti art as it can be seen today. The cross-pollination between graffiti and other areas in graphic design and illustration has always been a constant (Arte, 2015). In turn, current instances of graffiti art can be seen on record covers, in advertisements and in fashion, as well as textured on the walls of computer generated environments in games and in movies.

The motivation for this thesis is both artistic as well as technical and practical. From an artistic standpoint, the work stems from my personal background as a graffiti artist and from my desire to develop a set of tools that facilitate the production of digital artworks with

[1] Chinese whispers is a game in which a message is transformed as it is passed from one player to another. This analogy was mentioned during a talk on graffiti art at the 2019 Tag Conference in Amsterdam.

a specific and personalised stylistic signature. From the technical and practical standpoints, I propose that the computational re-creation of graffiti's main stylistic features is a valuable addition to the toolset of digital creatives, which makes the methods developed in this thesis generally useful tools across the 2D computational design space. Graffiti art is an intrinsic element of many contemporary urban-landscapes. As a result, an automatic graffiti generation system is a useful addition to procedural content generation (PCG) pipelines and is likely to contribute to a richer and more realistic rendition of computer generated urban environments in video games as well as movies. Finally, as suggested by Hertzmann (2010), I argue that the computational study of an art form implies making hypotheses about the processes involved in its creation, which establishes the ground for the development of a theory of art that is *generative* as opposed to only being descriptive (Hertzmann, 2010).

1.1 A short overview of graffiti styles

To build a more precise and more visual context for the methods that will be discussed in this thesis, it is useful to briefly cover some of the main aspects that characterize this art form and its different instantiations. Since its beginnings, graffiti has evolved into a multitude of styles, which differ depending on the individual artist, but also depending on a geographic location, visual culture, or to a required speed of execution.[2] In the following list, I will limit myself to a description of some fundamental styles in the category of the most traditional and influential form of graffiti art, such as could be seen in the 1980s on the streets and transit system of New York City.

1.1.1 Tags

In its most elementary, but fundamental, instantiation, graffiti art takes the form of a rapidly executed and highly stylised signature that conveys the artist's identity, style and skill. This form of graffiti is called a *tag*, which is the same term also used to denote a graffiti artist's pseudonym. The act of executing a tag is commonly referred to as *tagging* or *bombing*, where the latter implies executing many tags illegally.[3] Tags are meant to be executed quickly and in great quantity, usually using a marker or spray paint; their visual quality strongly depends on the speed and spontaneity with which they are executed. The manner in which a tag is written is commonly referred to as "handstyle" (Ferri, 2016) and identifies the artist's personal style and skill. A well executed handstyle is the result of years of practice, and its visual quality is directly related to the spontaneity in which the movements are executed. This is reflected in graffiti jargon with the term *flow*, which denotes the quality of execution of a tag.

With experience, tags are written more rapidly, and the stylised gestures involved in their creation are interiorised, becoming "muscle memory", which ultimately results in more fluid

[2] For example, depending on the local tolerance of the authorities to the art form.

[3] It should be noted that this study involves the stylistic elements of graffiti and not the social, legal and political implications.

Figure 1.1: Examples of tags. From top-left: TWISTER, AMAZE, EKSER, many tags in London, CRUIZER, CANSER, ENS. Most of the images are courtesy of `https://www.instagram.com/handstyler/`.

movements and spontaneous forms. The assimilation of these gestures is such, that it also influences a graffiti artist's handwriting, which is often recognisable with distinctive traces that resemble the ones of a tag .[4] The resulting handstyle becomes a personal stylistic signature of the artist, which is often transmitted to the curved portions of more complex forms of graffiti lettering (Ferri, 2016).

A note on tags and calligraphy. Calligraphy is the "art of beautiful writing" [5] and graffiti art is sometimes referred to as "urban calligraphy" (Arte, 2015), a term that is particularly appropriate for the case of tags. Graffiti artists Mode 2 (Craveiro, 2017) and Dado (Ferri, 2016) both compare the practice of writing tags to "shodo", the ancient art of Japanese calligraphy. Indeed, it is a persistent practice, and the resulting quality of a movement, that ultimately determines the visual quality of both calligraphy (Briem et al., 1983) and graffiti tags (Ferri, 2016). Because of this close relationship between tags and calligraphy, I will generally refer to both their respective traces as "calligraphic".

[4] I can speak from experience as this is also the case for my own handwriting.
[5] The term calligraphy originates from the Greek words *kallos* (beauty) and *graphein* (to write)

Figure 1.2: Examples of different steps of preparation of a graffiti piece. Left: a preliminary sketch is usually made with a pen or marker on paper. Middle: the sketch is transferred to a surface. Right: it is finally filled, colored and outlined.

1.1.2 (Master-)Pieces

The first instances of tags were written with a marker and often consisted of a name followed by a number, indicating for example the borough to which the writer belonged.[6] With the competition to gain visibility, graffiti writers started to use spray paint instead of markers and outlines were traced around the thickened strokes of the tag.[7] The tag slowly evolved into what is known as a "piece", a term that is used as an abbreviation for "masterpiece". A nice conceptual analogy of the transition from *tag* to *piece* is given by Italian graffiti writer Alessandro "Dado" Ferri (Ferri, 2016), who notes that in a piece, the writing sign of a tag becomes a "de-sign", resulting in a combination of forms that nevertheless recalls the structure and fluent gestures that compose a tag.

Many graffiti styles can be described through a composition of basic building blocks that are combined or fused to reveal the stylised outline of one or more letters. These building blocks vary from elongated geometric forms often referred to as "sticks", to more rounded forms known as "softies" that are traced with gestures similar to the ones that would be used for a tag (Figure 1.3). These components are often combined with other decorative elements, such as stylised arrows, stars, hearts and bars or "doodads", which are used to customise letter forms, to improve compositional balance, or to increase the senses of *dynamism* and *movement* in the piece. According to New York writer Rammelzee, the letter is "armed", preparing it for combat. All these visual elements are usually combined in ways that are evocative of an abstracted three dimensional composition, with overlapping, looping and intertwined/interlocking parts (Figure 1.4) and with extrusion effects that do not necessarily follow the strict rules of projective geometry.

The procedure used to design a graffiti piece usually starts from sketches made with a pen or a marker on paper (Figure 1.2, left). This is where a writer typically practices and studies different combinations and stylisations of letterforms, which are then transferred as

[6] The first tags are attributed to "TAKI 183", a foot messenger in New York who lived on 183rd street.
[7] The first example of this approach is attributed to the early graffiti artist "Super Kool 223" (Arte, 2015).

Figure 1.3: **(a)** Examples of a stick (top, ending with an arrow) and a softie (bottom), and two pieces by EGS (Helsinki, Finland) featuring mostly sticks **(b)** and softies **(c)**. The graffiti images are a courtesy of the artist.

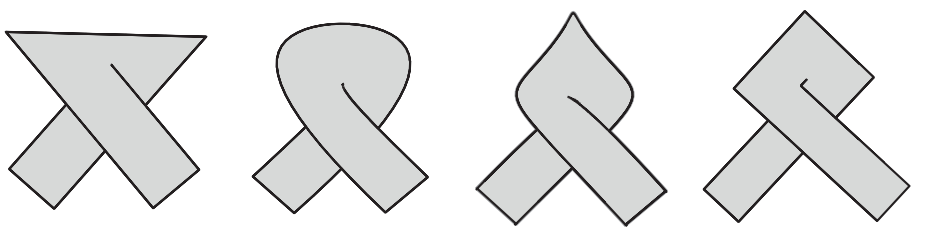


Figure 1.4: Self-overlapping loops that can be seen in graffiti pieces. This is a reproduction of examples given by Ferri (2016) , who calls this an "adjacency" form function. The loops are created with the stroking method presented in Chapter 6.

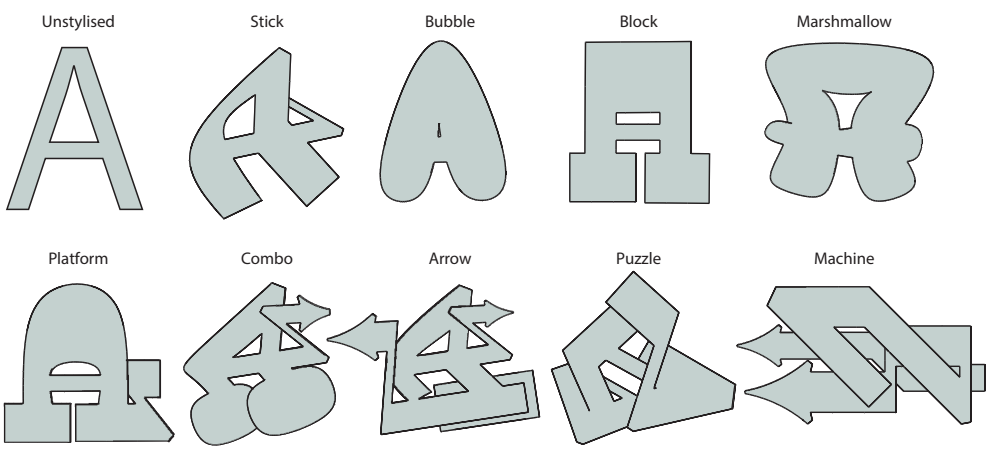


Figure 1.5: Examples of the letter "A" in an interpretation of the fundamental styles as described by (Ferri, 2016). The "unstylised" is a prototype letterform. Also these examples have been created with a rapid point and click procedure using the method presented in Chapter 6.

a larger sketch made with spray paint on a surface (Figure 1.2, middle). The sketch and (at times) also the background are then filled with one or more colors and more or less complex decorative effects. Finally, a final outline is traced to reveal the stylised letterforms. Often additional effects as extrusions and highlights are added to the outline for additional visual impact (Figure 1.2, right) .

Each graffiti writer develops their personal and often immediately recognisable style (Figure 1.6). However, many works of graffiti can be traced back to a number of stylistic "models" (Ferri, 2016), many of which were introduced in New York between the 1960s and the 1980s. Ferri (2016) categorises nine such styles (Figure 1.5) that vary from the simple "stick" style, to the rounded "marshmallow" style, to the mechanical looking "machine" style. He has analysed these styles through a combination of "compositional" and "form" functions that are listed in Appendix B. Some of these functions are indirectly implemented with the methods described in this thesis. Ferri also makes the example of an "unstylised" style, consisting of prototypical letter forms such as the ones that would be seen in a font such as Arial or Helvetica. Arte (2015) goes further by identifying the influence of specific typefaces on the origin of certain graffiti styles.

The most extreme form of graffiti stylisation is known as "wild style", in which most of the stylistic elements discussed above can be used together, resulting in highly abstracted letterforms. Letters are distorted, fragmented and interlocked in complex ways, often to the point of becoming unreadable to the untrained eye. It can be argued that graffiti pieces are particularly well suited to be studied through computational techniques. As a matter of fact, some forms of graffiti stylization are themselves inspired by technology and computer generated imagery. For example some pioneering forms of Wild Style were characterized by mechanical looking forms and where given names as "Mechanical Style" or even "Computer Rock" (Cooper and Chalfant, 1984).

1.1.3 Other graffiti styles and elements

The description above is a simplified overview of a much more complex taxonomy of styles, which have evolved throughout the development of graffiti art. For an exhaustive and in depth categorisation of different graffiti styles, together with an analysis of their structure and development, I suggest the excellent treatise of Ferri (2016) and Arte (2015). In the context of this thesis, and for a general understanding of the graffiti that can be typically seen in a city, it is useful to briefly summarise a few more stylistic categories: throw ups, puppets and abstract.

Throw ups. A rapidly executed and often rounded outline representation of the tag, often also quickly filled in with a single colour. While also intended to be executed rapidly and in great quantity, throw-ups are perhaps the most perceptually complex form of graffiti art. Letters are abstracted to their essence and executed with a minimum number of highly stylised

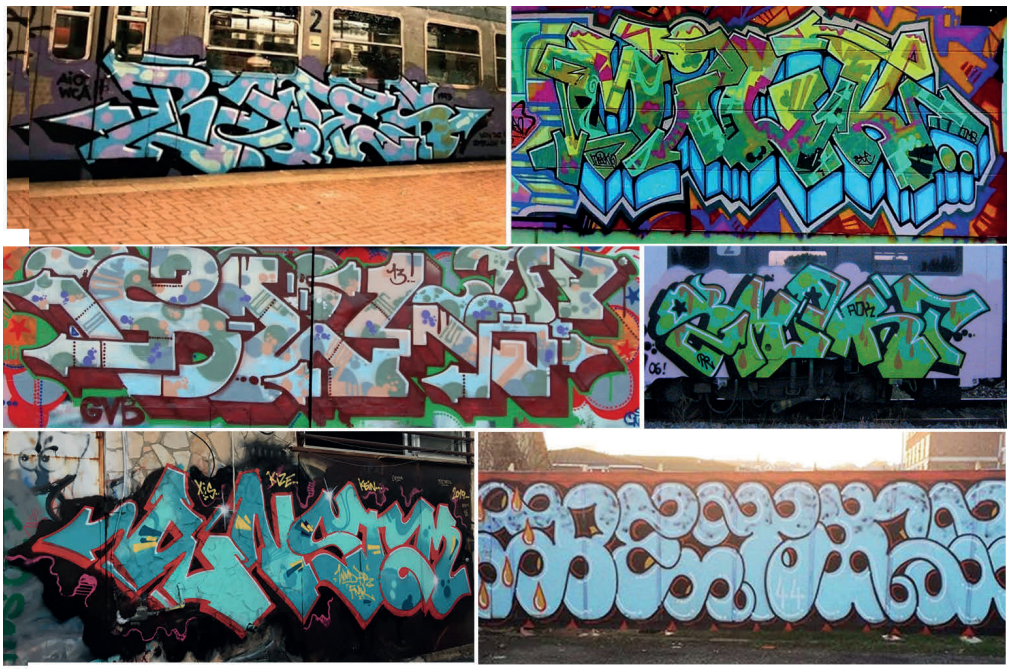


Figure 1.6: Graffiti. *First row:* BATES (Copenhagen, Denmark), MILK (Munich, Germany); photos are courtesy of `https://www.instagram.com/johnnation/`. *Second row:* SEL (Haarlem, Netherlands), SMART (Florence, Italy). *Third row:* ENS+STAM (Florence, Italy), PETRO (London, UK). Photos are courtesy of the artists or taken by me.

strokes, while maintaining their legibility (Figure 1.7).

Figure 1.7: Examples of throw-ups: DUEL (NYC, USA), SM(ART) (Florence, Italy), GREY (SF, USA). Photos courtesy of the artists.

Puppets. Pieces are often accompanied by caricatures of animals or persons that are commonly known as *puppets*. Often these caricatures are executed in a stylised manner, with the same gestures, forms and visual conventions that characterise stylised graffiti letters (Ferri, 2016). Sometimes that stylisation is such that the puppet also follows a structure similar to a letterform (Figure 1.8, left).

Figure 1.8: Examples of puppets: POPZ 100 (Nottigham, UK) and CMP (Copenhagen, Denmark), courtesy of `https://www.instagram.com/johnnation/`; ENS (Florence, Italy).

Abstract Style. In some cases the same stylistic elements that are used in Wild Style or tags can be combined freely, without following the structure of a letter. This results in an abstract composition that has recognisable aesthetic properties, similar to graffiti, but does not have any specific representation of lettering (Stowers and Goldman, 1997) (Figure 1.9).

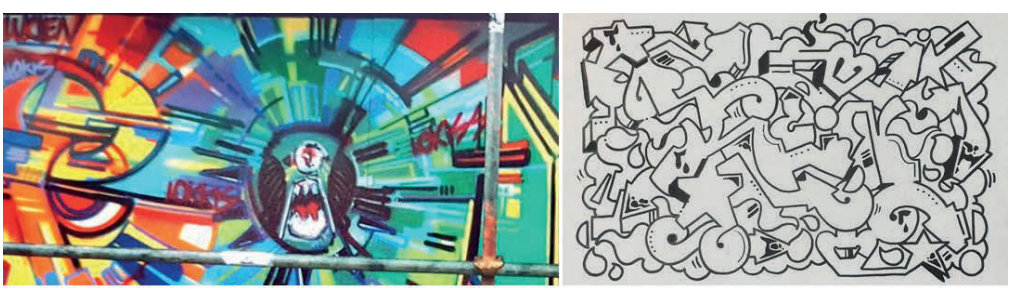


Figure 1.9: Examples of abstract styles. *Left:* LOKISS (Paris, France); courtesy of `https://www.instagram.com/johnnation/`. *Right:* KEIN (Florence, Italy), courtesy of the artists.

1.2 Graffiti in the Digital and Virtual Realms

With its proliferation, graffiti art has become increasingly present in cities around the globe. This presence has become also digital, with traces of graffiti appearing in a variety of computer generated content ranging from graphic design to urban scenes in games and in movies.

1.2.1 Graffiti in Graphic Design

Arte (2015) draws a parallel between the development of certain graffiti styles and the popularity of typefaces at the same time and geographic location. With time, this influence has become mutual and the visual elements of graffiti art can be found in instances of typographic

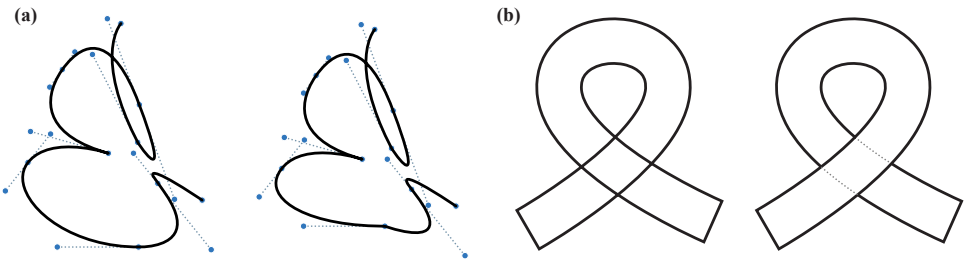


Figure 1.10: Editing graffiti with conventional vector graphics techniques. **(a)** Attempt at producing a tag-like letter "E" with Bézier curves. The procedure requires placing control points at locations that are not easily related to the movement that would be executed to produce the trace (left). Editing the location of a control point is likely to result in a curve that does not appear natural (right). **(b)** Attempt at recreating a loop, similar to the ones shown in Figure 1.4. Reproducing the overlap requires manually removing the dashed segments on the right.

and graphic design (Craveiro, 2017), as well as advertisement, record designs and fashion. Indeed, a number of contemporary graphic designers and typographers come from a previous practice-based experience in graffiti art (Craveiro, 2017; Kimvall, 2007).

Vector graphics software is often the tool of choice when it comes to designing typography or graphics for print. However, designing graffiti or calligraphy, can be challenging with the conventional tools of Computer-Aided Geometric Design (CAGD). Tags for instance, are the result of highly skilled and well-practiced movements. Reproducing their traces usually requires either (i) an equally skilled movement executed with a digitiser device or (ii) the careful selection of a large number of curve control points, the location of which can be highly unintuitive (Figure 1.10.a). A similar issue is observed by font designer Charles Bigelow [8] for the case of fonts that mimic calligraphy or handwriting; he notes (Wang, 2013):

> "I am sometimes sorry to see that the spirit and grace of the moving hand and tool, whether pen, brush, or reed, are lost in modern typographic technology, but now that the basic problems of outline font technology are solved, perhaps someone in the future will work on restoring the human action."

Similar challenges hold also for the design of graffiti pieces. In particular, conventional drawing applications usually assume a strict back-to-front ordering of geometry (Figure 1.10.b). As a result, reproducing the self-overlaps and interweaving that characterises graffiti stylised letters, often requires subdividing a design into a number of unintuitive parts. This results in a "vector soup" that is often difficult to edit and to manage. Ultimately, these

[8] Charles Bigelow has designed a number of widely used digital fonts, among which Lucida, which is used also in this document to typeset equations

Figure 1.11: Examples of graffiti in the videogame GTA. On the left, GTA V, with a throw-up by NYC graffiti artist COPE2 (stylistically high quality) textured in different parts of the virtual environment. On the right, interactive tagging session in GTA Sant Andreas.

issues hinder the creative design process, thus not always justifying the expression "computer aided design".

1.2.2 Graffiti in Games and Movies

In contrast with more traditional art forms, graffiti unfolds on the surfaces of an urban environment. As a matter of fact, today graffiti art can be considered as one distinctive feature that characterises urban landscapes around the globe. Likewise, the same feature is also often present on the textured surfaces of buildings and objects that can be seen in computer generated urban scenes, in games as well as in movies.

Graffiti content for games is usually prepared by texturing photographs or by specifically commissioned graffiti art (real, or painted on the computer with tools such as Adobe Photoshop). This method has various limitations, as the amount of work needed is directly proportional to the size of the game world, and the graffiti images have a fixed resolution that is constrained by memory limitations. An illustrative example of this approach can be found in the Grand Theft Auto game series, which features one of the largest in-game urban environments to date and extensively uses graffiti as an aesthetic in-game element and landmark feature. Although graffiti art is an intrinsic in-game element of the game series and its quality is greatly improving across releases, we can still notice a limited variety of low resolution tags and pieces which are repetitively textured across the walls of the game (Figure 1.11, left). This repetition is unrealistic, since human movements are characterised by an intrinsic variability (Harris and Wolpert, 1998) and so are the traces of multiple tags or drawings. The game player is also enabled to produce their own graffiti (Figure 1.11, right), but this process is also highly unrealistic, with the image of a piece slowly fading in while the virtual character performs random-looking arm movements.

Figure 1.12: Left, computer generated graffiti in the movie *Baby Driver* (2017). Middle, human made alien graffiti in the movie *Alien Nation* (1988). Right, hypothetical Klingon graffiti for the word "writing" generated with the system described in this thesis.

Similar points hold also for procedural content generation (PCG) in movies. One illustrative example is the movie *Baby Driver*, in which the introductory titles, as well as many movie scenes, feature computer generated graffiti with sentences that pertain to the plot of the movie (Figure 1.12, left). In the context of science fiction, the 80s movie *Alien Nation* featured instances of "Tenctonese" graffiti, made by a race of alien visitors stranded on earth (Figure 1.12, middle). In a hypothetical movie production setting, one could then wonder: what would graffiti look like if it were made by Klingon[9] visitors instead? A suitably built procedural content generation system can provide an automatic answer to this kind of question (Figure 1.12, right). Furthermore, such a system has the potential to work with procedurally generated and even extremely large environments for which the production of realistic and convincing graffiti textures would require a prohibitive amount of memory storage and human resource.

More generally, the development of PCG systems capable of creating, or evaluating, content with a specific style, let it be architectural, artistic or other, is still an open area of research for the PCG community (Togelius et al., 2013). Similar challenges hold for the area of Non Photorealistic Rendering and Animation (NPAR) (Gooch et al., 2010), a subfield of computer graphics that is aimed at the simulation of artistic techniques and styles and at clarity of representation (Kyprianidis et al., 2013). As a result, graffiti art offers an interesting testbed for techniques that are relevant to both the domains of computer graphics and PCG: it is an artistic process and style that can be simulated using techniques that are relevant to the NPAR community, but it develops and unfolds in an urban environment; therefore it is highly suitable for being generated and studied in the context of virtual 3D environments.

1.2.3 Computer Aided Graffiti Design

Recently, some work has been done at the intersection of graffiti art with technology. Cassidy Curtis (2002) developed the *Graffiti Archeology* project, an online platform that visualises a timelapse of graffiti being painted on a selected number of walls. Jurg Lehni has developed

[9] An alien race in the *Star Trek* movie series

HEKTOR, a Cartesian drawing machine that is able to mechanically trace vector images with a spray can on a wall (Lehni, 2004). The Graffiti Analysis project by Evan Roth has resulted in a series of low cost motion capture devices that allow to easily record or digitally reproduce the gestures done during tagging (Roth et al., 2009). This has resulted in collectives of hackers and artists known as *Graffiti Research Lab*s (Keough, 2010) that develop technologies that enable the materialisation of graffiti with do-it-yourself (DIY) devices. One spinoff of this project has resulted in the *EyeWriter* system (Tempt1 et al., 2009), which has allowed a paralised graffiti artist *TEMPT1* to create graffiti with the movements of their eyes. New York graffiti artist KATSU (Holland Michael, 2015), and others (Vempati et al., 2018), have developed drones that can reproduce large scale graffiti with spray paint on any surface. While these methods provide innovative ways to materialise digitised traces with the medium typical of graffiti art, they do not actually provide computational means to generate traces that resemble this art form or to assist their generation within a CAGD pipeline.

With the methods developed in this thesis, I seek to address this gap, as well as some of the previously posed challenges, by:

1. Developing a set of primitives that can be combined to rapidly define curves and designs that are similar to calligraphy or to graffiti art.

2. Developing an interface for editing these primitives, which is similar to the one conventionally used curves in CAGD applications, and with a representation that is intuitive to edit and to manage.

3. Developing a system where a user is able to insert a string of text in a language of choice, to reproduce the string with a graffiti stylised output, to render the output at arbitrary resolutions and to edit the output with the same user interface introduce above.

The system should then:

- Produce content that can be varied in ways that reproduce the variability that can be observed in multiple instances of drawing or writing made by a human.

- Produce content that intrinsically captures a plausible sequence of movements, similar to the one that would be followed by a human when producing an artwork. This enables the generation of realistic animations of the graffiti production process and its reproduction with robots or other fabrication devices.

To this end, I propose a *two-level* hierarchical approach to synthetic graffiti generation. The first level defines a set of *stroke primitives*, building blocks that can be easily varied and combined to reproduce different kinds of graffiti stylised letters and compositions. The second level, recovers these primitives from existing geometric inputs, such as traces made with

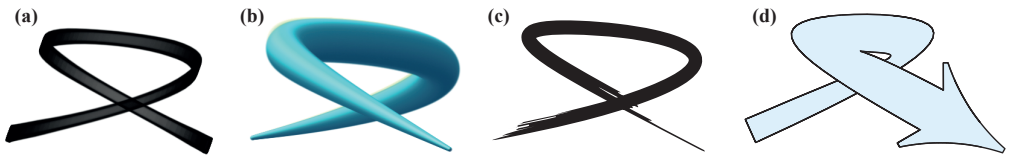


Figure 1.13: A few different kinds of strokes.

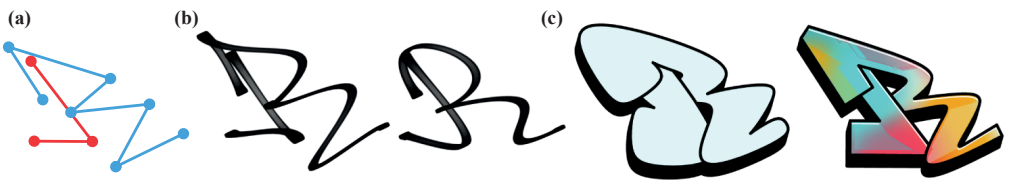


Figure 1.14: Example stroke stylisations of a motor plan for the letter "R" **(a)**. **(b)** two calligraphic stylisations of the motor plan. **(c)** two outline stylisations of the motor plan.

a digitiser device or the outlines of a font. These geometric inputs then become a seed for the genesis of different kinds of graffiti stylisations. The thesis is similarly organised in *two parts* that are outlined in the next two sections.

1.3 Part I: Graffiti primitives

Some of the previously discussed stylistic elements of graffiti already suggest their computational counterparts. For example, there is a wealth of computational models of human arm and hand movements to choose from, when it comes to mimicking the skilled gestures that are used to produce a tag. Also, a reader that is familiar with 2d computer graphics techniques is likely to notice a similarity between fundamental graffiti building blocks such as the "stick" or the "softie", and well established vector based stroking methods such the skeletal strokes method of Hsu and Lee (1994).

In order to treat different graffiti stylisations with a common representation, I propose one simple guiding concept, that is shared by all the methods developed in this thesis: Different stylisations of a letterform can be described with a bi-level representation, consisting of a *structural* and a *stylistic* component.

The *structural* component is a schematic representation of a letterform in terms of a *motor plan* consisting of a sparse sequence of vertices connected by polylines. It describes the layout and order of a series of drawing movements that can be used to trace the letterform. In practice, the polylines used are similar to the control polygons typical of CAGD applications, and thus both can be specified and edited in a similar manner.

The *stylistic* component consists of parametric stroke primitives that are constructed along a motor plan and produce different letterform stylisations. A stroke can embody the

trace of ink or paint left with a calligraphic gesture, or a depiction of such a trace in the form of an image or a stylised outline (Figure 1.13). A *stroke stylisation* of a motor plan consists of a series of strokes constructed along the vertices of a motor plan (Figure 1.14). Part I of this thesis describes a set of stroke primitives that result in stroke stylisations that resemble calligraphy and graffiti art. For our use case, I categorise stroke stylisation into two types: *calligraphic stylsation* (Figure 1.14a) and *outline stylisation* (Figure 1.14b). The first can be used to depict tags and the second to depict pieces.

1.3.1 Calligraphic stylisation: Movement and tags

A tag can be considered as an elementary "atom" of graffiti art and the way in which it is written often impacts also the appearance of other forms of graffiti stylisation. It follows that tags are also a suitable starting point for the development of the proposed set of graffiti primitives. That being the case, let's briefly examine some examples of tags, for which artist Evan Roth has conveniently isolated a number of letters of the alphabet, within his project "Graffiti Taxonomy" (Roth, 2011). Each letter instance is made with a distinct handstyle, and within each letter it is possible to identify subsets that share a similar underlying structure, but are yet executed with clearly different handstyles. For example, in Figure 1.15a it is possible to observe a number of capital "N"s, a number of lower case "n", where one such case is equivalent to a mirrored *И*.

Considering each of these subsets in isolation, leads to a concept that I call "style as kinematics", in which a number of different stylisations of a letterform can be seen as variations of a movement that follows a common motor plan. This is consistent with the previously introduced bi-level representation, with the stylistic component becoming a *kinematic component* that describes the hypothetical trajectory of a writing tool.

To implement this concept, I propose a *movement centric* approach to curve generation, in which a curve is defined through a physiologically plausible simulation of a (human) movement underlying its production rather than by an explicit definition of its geometry. The implementation relies on the systematic application of methods and principles from the domains of handwriting analysis and synthesis (Plamondon et al., 2009), computational motor control (Rosenbaum, 2009) as well as robotics (Calinon, 2016b). I argue that, with an appropriate parameterisation, some of these methods become a useful tool to generate tags with a procedure that is similar to established CAGD methods, but with the additional advantage of capturing both the geometry and kinematics of a human made trace with a single integrated representation. Chapters 4 and 5 demonstrate two different but complementary methods that can be used to implement this procedure and to automatically generate strokes that closely resemble the ones that would be seen in tags or calligraphy produced by human experts.

Chapter 4 presents an extension and an application of the Sigma Lognormal model, a

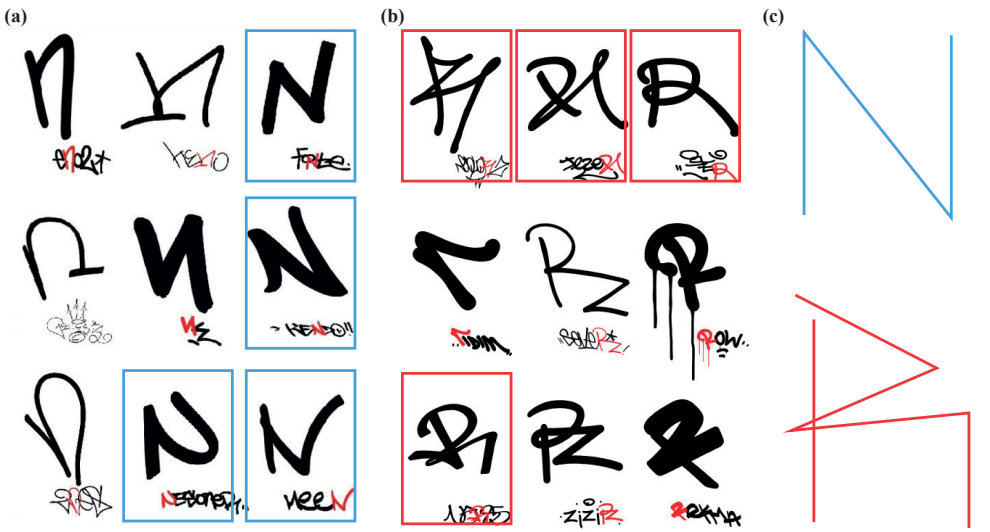


Figure 1.15: Common letter structures. Some examples of the letters "N" **(a)** and "R" **(b)** isolated from tags by Evan Roth in his graffiti taxonomy project. **(c)** Two motor plans that can be used to characterise the emphasised letters in **(a)** and **(b)**. Image courtesy of Evan Roth.

model of handwriting movements that is widely used for handwriting analysis (Diaz-Cabrera et al., 2018) and synthesis (Plamondon et al., 2014) applications. This model describes handwriting movements with a set of parameters that have a well defined physiological interpretation. As a result, perturbations of these parameters result in variations of a trajectory that resemble the ones that can be observed in real instances of handwriting or drawing. The resulting trajectories are kinematically similar to the ones produced by a human; this chapter also demonstrates how this property can be exploited to generate convincing stroke renderings and animations.

Chapter 5 presents a similar interface with an approach based on optimisation, in which a user *explicitly defines* the desired variability/precision of a movement. The method produces a distribution of trajectories rather than a single one and the trajectories are generated with an optimisation, which rewards a tradeoff between the required precision and smoothness of a movement.

These two methods can be used to generate and stylise strokes that appear hand drawn, while giving a high level of parametric control to the user. I argue that this approach has, by definition, a series of useful properties such as:

- Providing a representation of variability that intrinsically built in the abstract representation of a pattern. This allows to reproduce the variations that are similar to the ones that can be seen in multiple traces made by one or more artists through simple pa-

Figure 1.16: Examples of a letter "N" isolated from various pieces.

rameter perturbations (Chapter 4) or stochastic sampling of a "trajectory distribution" (Chapter 5). This property can be exploited for artistic/design purposes, to but also for data augmentation in machine learning applications.

- Generating kinematics that are similar to the ones that would be observed in a human movement. This can be exploited to generate realistic stroke animations and natural looking animations for virtual characters or robots/fabrication devices (Berio et al., 2016) that embody the drawing process, but also to produce more realistic renderings of artistic traces (Chapter 4).

- Providing a flexible and high-level feature representation. This representation can be exploited (i) to simplify procedural generation methods (Chapter 5) and (ii) for the implementation of data-driven stylisation methods using generative models (Berio et al., 2017a) that can be trained with fewer samples (due to the possibility of easily generating augmented data) and the output of which can be manipulated in a meaningful manner (e.g. through parametric manipulation and user interaction).

The proposed movement centric approach to curve generation adopts an "embodied aesthetics" hypothesis (Freedberg and Gallese, 2007). This maintains that the observation of static marks or traces left by a drawing movement activates motor areas of the brain (Freedberg and Gallese, 2007; Longcamp et al., 2003), inducing an approximate mental simulation of a likely generative movement (Freyd, 1983; Pignocchi, 2010; Leder et al., 2012). It is also hypothesised that such a brain stimulation can influence the aesthetic appreciation of the static trace (Leder et al., 2012). This leads to the conjecture that a similar phenomenon should occur with synthetic traces produced with an appropriately simulated movement.

1.3.2 Outline stylisation: Parts and pieces

A similar observation to the one made for tags, can also be made for the stylised letter outlines of graffiti pieces (Figure 1.16). In this case, different stroke stylisations of a motor plan can be viewed as an *assembly of parts* that are fused to produce an outline. As previously mentioned,

these parts are often overlapped and intertwined in a way that is difficult to reproduce as a simple back-to-front composition.

Chapter 6 presents a variant of the skeletal strokes algorithm (Hsu and Lee, 1994) that is especially built to reproduce these characteristic features of graffiti art. Similarly to the previously defined methods, a user can define a stroke with a sparse sequence of points, but this method results in a stylised outline rather than a single trace. Variably smooth outlines are produced by also synthesising a movement, specifically with the same optimisation-based method that is used Chapter 5 to reproduce tags. This results in a parameter space that covers a variety of different graffiti strokes and styles.

The outlines of one or more strokes can be combined interactively with local union, layering and self-overlap effects, while always maintaining the underlying stroke structure. The method then produces vector output with no artificial artwork splits, patches or masks to render the non-global layering, where each path of the vector output is part of the desired outline. This results in an output that can be reproduced on a screen, but also with robots or other kinds of fabrication devices. Finally, the output can be rendered with different fill-ins, highlights and decorative effects that resemble the ones that can be often seen in graffiti pieces.

1.3.3 Overall contributions of Part I

The first part of the thesis results in a set of parametric stroke primitives that facilitate rapid authoring of graffiti-stylised letters and designs. These primitives are specified by means of a motor plan, a simple structural representation consisting of sparse sequences of points and thus easy to edit with a point and click procedure. Different stylisations of the motor plan are then produced through parametric variations of the stroke primitives. This representation is also helpful when used in combination with procedural generation methods, since the procedures are left with the simplified task of generating sparse sequences of points, which can be subsequently transformed into a variety of stylised outputs.

These methods are conceived with the goal of reproducing instances of graffiti art or calligraphy. However, the resulting contributions extend to the more general spectrum of artistic applications of CAGD, namely:

(a) A movement centric approach to curve design, aimed at generating, animating and rendering traces that appear written or drawn by hand.

(b) A set of geometric primitives with a built-in representation of variability, which can be used to produce variations similar to the ones seen in multiple instances of drawing/writing made by a human.

(c) A novel vector-based stroking method designed to enable strokes with self-overlaps and non-global layering effects.

1.4 Part II: Recovering graffiti primitives from geometry

With a set of stroke primitives in hand, the second part of the dissertation is focused on a "reverse" procedure with respect to the first: recovering plausible combinations of stroke primitives that reconstruct a given input geometry. This procedure transforms unstructured geometric inputs into a parametric representation that thus enables all the functionalities that are discussed in part I of the thesis. The process of procedural graffiti generation is thus transformed into one of *parametric stylisation*, with a range of geometric inputs serving as source of possible structures that can be stylised and transformed into different kinds of graffiti.

1.4.1 Geometric input analysis

I distinguish two broad categories of geometric data and two corresponding datasets, which I consider particularly useful for the task at hand of graffiti content generation:

The first type of geometric data consists of tags, handwriting, drawing, or arbitrary vector paths that are digitised as ordered sequences of points.[10] One particularly useful dataset of this kind is known as the "Graffiti Analysis Database" (Roth et al., 2009); the database contains thousands of tags, recorded with a variety of devices ranging from tablets to DIY marker-based motion tracking systems. The database provides data that is already a potentially useful source of graffiti content. However, the data has varying sampling quality and the point-sequence format is difficult to modify or edit in any meaningful way.

The second type of geometric input consists of the outlines of one or more *glyphs*.. A glyph is usually a letter of the alphabet or a readable symbol of a given writing system, but it can also represent other kinds of "non-letter" objects such as a silhouette or an abstract pattern. A collection of glyphs with a consistent style is usually referred to as a *font*. Using glyph outlines as a geometric input results in an approach that is consistent with the idea proposed by Arte (2015) that the origin of certain graffiti styles can be related to specific fonts. Publicly available fonts provide an almost inexhaustible source of letterforms, in a variety of languages and writing systems, and with a variety of different styles and structures. Recovering a structural representation from such outlines is not trivial (Wang, 2013). However, recovering one that is compatible with the parametric methods described in Part I enables the generation of graffiti stylisations with an equally varied structural and stylistic range.

In order to treat this diversity of inputs, I propose an approach that is grounded on well-studied principles of visual perception and that relies on a geometric analysis only, thus not requiring training data or predefined templates. The analysis procedure builds on a set of "Curvilinear Shape Features" (or CSFs): descriptors of concave and convex shape features that are built from local symmetry axes and have an associated region of support . This feature representation is introduced in Chapter 7 and it is subsequently used as a basis for all the remaining methods in Part II of the thesis.

[10] Commonly known as *online data* in the handwriting analysis literature.

1.4.2 Trace based methods

Chapters 8 and 9 are focused on geometric inputs of the first type, that is traces of various kinds, which can be used to generate variations and stylisations of tags.

Chapter 8 uses CSFs to reconstruct traces in terms of strokes parameterised with the Sigma Lognormal model. The method purposely ignores the kinematics embedded in the input, allowing it to handle geometry where this information may be absent or degraded due to poor digitisation quality. The resulting reconstruction thus infers physiologically plausible kinematics from geometry and results in a concise and meaningful representation of a movement that extends the rendering, animation and parametric variation methods described in Chapter 4 to arbitrary traces.

Chapter 9 uses this reconstruction to drive an example-based stylisation method based on a recurrent neural network. The method allows to stylise an input structure in way that resembles an example trace, or to transfer qualities of movement between different traces. This results in a novel form of "style-transfer" that is similar to existing curve-based methods in computer graphics (Hertzmann et al., 2002; Li et al., 2013) but is based on the proposed concept of "style as kinematics". With the pretest of stylisation, this chapter also investigates the utility of a high-level representation of movement when combined with recently popular deep learning approaches. The results demonstrate that the additional structure provided by the Sigma-Lognormal model can be exploited to train a data-hungry method with as few as single training example.

1.4.3 Outline based methods

The trace-based methods in Chapters 8 and 9 allow the generation and rendering of tags while enabling realistic variations and animations, which effectively addresses some of the issues previously raised for graffiti content generation in games. However, for a procedural graffiti generation system to be useful, it is certainly desirable for a user to specify a string of text and then render it with different graffiti styles.

In order to achieve this, Chapter 10 presents a method that partitions the outlines of a glyph into a set of potentially overlapping and intersecting strokes , augmented with semantic annotations that determine connectivity relations or features such as corners. The output of this procedure initially reconstructs the outline as precisely as possible. Then, these strokes can easily be transformed into a motor plan that is compatible with all the other methods developed in the thesis, thus enabling all the previously discussed stylisation and rendering methods.

Chapter 11 demonstrates a number of examples of this stylisation procedure, and how the outlines of a font can be used to generate instances of calligraphy and graffiti art in a variety of different styles, structures and languages. The stylised outputs are expressed as parametric stroke primitives, which can be edited, varied and rendered at arbitrary resolu-

tions, with exactly the same techniques that are presented in Part I of the thesis.

1.4.4 Overall contributions of Part II

The methods described in part II of the thesis, extend the functionalities described in part I to arbitrary geometric data. This results in a graffiti content generation system that can be used to (i) generate and render realistic tags with variations that resemble the ones that would be seen in real instances of graffiti and (ii) to generate graffiti stylised strings in arbitrary languages and writing systems, with a form and structure that is derived from the outlines of a font.

These methods, are conceived to work principally with geometric data that represents letterforms, such as handwriting traces or glyph outlines. However, a wider range of inputs is possible and this potentially transfers the notion of graffiti stylisation beyond letters, in a process that I refer to as *Graffitization*. With the aid of robotic and digital fabrication technologies, this concept is not only applicable to the virtual, but extends back to the material world.

Similarly to Part I of this thesis, the context of graffiti results in a number of contributions that extend to a larger application domain, namely

1. A representation of concave and convex curvilinear shape features, with wider applications in shape analysis problems.

2. A novel method that reconstructs Sigma Lognormal parameters from the geometry of a trace, which extends existing methods that all require high-quality input kinematics (Plamondon et al., 2014; Ferrer et al., 2018) and with consequent applications to pattern recognition for handwriting analysis.

3. An example-driven curve stylisation approach that is similar to existing methods (Hertzmann et al., 2002; Li et al., 2013) but takes movement kinematics into account.

4. A segmentation method, that decomposes font outlines into constituent strokes without requiring training data, which is generally useful for type-design (Wang, 2013) and animation (Gingold et al., 2008) applications.

1.5 Publications

The chapters of this thesis discuss work and solutions that I have personally conceived and implemented during my PhD studies. However, most of the work reported and published involves various collaborations. Most chapters begin with a preamble that lists the associated publications and explicitly states my contributions and those of my collaborators where needed. All these publications have been peer-reviewed and accepted, with the exception of the material in Chapter 10 and a portion of Chapter 7 which are part of a journal submission

that is in preparation at the time of writing. The thesis extends and develops previously published work further, with additional details and with subsequently developed refinements or extensions. For the complete list of publications please refer to Appendix A. A lot of the material in this thesis, has resulted in the development of software, which is best documented through videos or interactive demonstrations. At the time of writing, a number of videos are available at the following web address: `https://www.enist.org/post/research/autograff/`. The videos are organised according to the chapter structure of this thesis, and the page is intended to be updated with upcoming source code releases and new supplemental material. The reader is invited to view the videos for a chapter if these are present.

Chapter 2

Notation and preliminary definitions

Vectors and matrices are denoted by bold symbols, with vectors always denoted by lower case symbols (e.g. $\boldsymbol{x}, \boldsymbol{\phi}$) and matrices by upper case symbols (e.g. $\boldsymbol{A}, \boldsymbol{\Phi}$). We adopt the Matlab-like indexing notation $\boldsymbol{x}_{a:b}$ or $\boldsymbol{X}_{a:b}$ to indicate the consecutive rows of a vector or matrix from a to b including b. Vectors are always assumed to be columns, however, like Murphy (2012), we use a notation of the type $\boldsymbol{x} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$ to denote vectors stacked along a column. In practice, this should be $\boldsymbol{x} = \left[\boldsymbol{x}_1^T, \cdots, \boldsymbol{x}_n^\top\right]^\top$, but we choose the former notation because it avoids clutter and the lower case symbol on the left clarifies that the operation results in a vector. Conversely, $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$ denotes a matrix, the columns of which are given by the vectors $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$.

We will use the Gaussian (or normal) probability distribution at different stages of the document and denote it with $\mathcal{N}$. Using bold symbols $\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ implies that the distribution is multivariate with vector mean $\boldsymbol{\mu}$ and matrix covariance $\boldsymbol{\Sigma}$. Conversely, $\mathcal{N}\left(\mu, \sigma\right)$ implies an univariate distrbution with scalar mean and standard deviation μ, σ. Other symbols and notations will be described when required.

2.1 Geometry

This thesis is mainly concerned with the generation of 2D images is vector form. For consistency with most 2D computer graphics packages, we will assume a *left handed* coordinate system with vertical coordinates increasing downwards and angles increasing clockwise.

Curves: A curve is a "trip taken by a moving point" (O'Neill, 2006); mathematically, a mapping $\gamma : I \rightarrow \mathbb{R}^D$ from an interval $I = [a, b]$ to its *trace* as a set of D-dimensional points in $\mathbb{R}^D$. A planar curve maps the interval to points in $\mathbb{R}^2$. A curve is simple if it does not intersect itself, and it is closed if $\gamma(a) = \gamma(b)$, such that its trace forms a loop.

Polyline: A polyline is a piecewise linear curve, consisting of a connected sequence of straight line segments. It is specified as a sequence of *vertices* with coordinates $\boldsymbol{p}_i$.

Trajectory: A trajectory is a continuously differentiable curve parameterized by "time", with coordinates denoted by $\boldsymbol{x}(t)$. The time derivatives of a trajectory are denoted as $\dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}$, for first and second order, and $\overset{(n)}{\boldsymbol{x}}$ for order $n > 2$. A *discrete trajectory* is a trajectory sampled at a discrete time step Δt and resulting in a polyline. The coordinates of the kth sample are denoted by $\boldsymbol{x}_k$, where $\boldsymbol{x}_k = \boldsymbol{x}(k\Delta t)$.

Curvature: Mathematically, the curvature [1] of a continuous curve is a differential measure of its deviation from straightness, i.e. it expresses the local change in tangent direction. For a planar trajectory, $\gamma(t) = [x(t), y(t)]^\top$, curvature can be computed with the classic formula:

$$\kappa(t) = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}} \quad . \tag{2.1}$$

Absolute curvature is the reciprocal of the *radius of curvature* $r(t) = 1/|\kappa(t)|$, corresponding to an *osculating circle* (or circle of curvature) sharing a tangent with the trajectory at $\gamma(t)$ and centered along the normal to $\gamma(t)$. For a curve sampled at a uniform distance step Δs, curvature can be approximated by:

$$\kappa_s = \frac{\phi_s}{\Delta s} \quad , \tag{2.2}$$

with s the distance along the curve and ϕ_s the *turning angle* between two successive trace segments. In the limit, as $\Delta s \to 0$, this approximation becomes the *arc length* or *unit speed* parameterisation of the corresponding curve, where the curvature function $\kappa(s)$ is known as the curve's *intrinsic* or *natural* equation. The function $\kappa(s)$ can be used to recover the trace of a plane curve up to Euclidean motions, that is up to translations and rotations (Abbena et al., 2017) with:

$$x(s) = \int \cos\phi(s)ds \quad \text{and} \quad y(s) = \int \sin\phi(s)ds, \quad \text{with} \quad \phi(s) = \int \kappa(s)ds \tag{2.3}$$

and where, for example, different kinds of spirals can be computed with $\kappa(s)$ being a monotonic function of arc-length.[2]

Solid objects, figure, ground, and outline: A solid object Ω, or object for short, is a subset of $\mathbb{R}^2$ bounded by a rigid outline $\partial\Omega$, a set of closed and simple curves called *contours*. Contours delimit points $\in \Omega$ (the object's interior, or *figure*) from points in the complement $\mathbb{R}^2 - \Omega$ (the object's exterior, or *ground*). Contours are oriented so that travelling along a contour the figure is always on the right. For example a doughnut shape has an outline consisting of two circular contours, an outer contour oriented clockwise, and an inner contour oriented counterclockwise.

Discrete trace and contour: *Discrete trace*, or *trace* for short, is a sequence of 2D points obtained from a sampling procedure. In this work, unless otherwise specified, we will use for simplicity a uniform sampling on the natural paramterisation of a curve , thus resulting in polylines with segments of approximately constant length Δs. When the trace points are sampled from a continuous curve, this sampling approximates the curve's arc-length parameterisation as $\Delta s \to 0$. In practice, the trace can be open or closed, and its points are not necessarily sampled from a continuous input. The input may consist of the samples produced from a digitization device such as a mouse or tablet, or edges obtained through an image filtering procedure. The input can also consist of a contour of an object outline, in which case we also refer to the resulting trace as a (discrete) contour, implying that it is guaranteed to

[1] Coolidge (1952) exposes an interesting history behind the mathematical definition of curvature. A modern definition in terms of calculus is attributed to Newton in the 1700s. However a related concept of "curvitas" can be traced back to the work of Nicolas Oresme in the 1300s.

[2] Refer to the text book by Abbena et al. (2017) for examples on how intrinsic equations can be exploited as a flexible curve generation tool.

be simple and closed. The coordinates of discrete contours and traces are both denoted as $\boldsymbol{z}_i$.

2.2 Motor plans and strokes:

The fundamental primitive used in this thesis is a *stroke*:

Stroke: A *stroke* is an elongated 2D region, for example obtained as the result of a drawing or painting gesture between two positions on a drawing surface. Computationally, a stroke can be represented in a tri-partite way as: (i) an axial curve or *spine*, (ii) a possibly non-symmetrical *width profile* that determines the local thickness of a stroke, and (iii) a *generator* that fills an area or creates an outline or fills an area as it traverses the axial curve. The generator can take different forms, such as a possibly non-symmetrical brush footprint that embodies the trace of ink or paint left with a calligraphic gesture, or an outline that is constructed along the spine.

Motor plan: Strokes are constructed with the analogy of a writing/drawing tool moving and leaving marks on a surface. The layout and order of traveral of strokes is defined according to a *motor plan*, a schematic reprsentation of one or more (drawing) movements. A motor plan, denoted as either P or Q, consists of a sequence of vertices $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_M$, with consecutive vertices connected by one or more polylines.

Kinematic realisation: It is a sequence of trajectories that follow the polylines of a motor plan. It is denoted with the script version of the motor plan symbols, $\mathscr{P}$ or $\mathscr{Q}$, and it is computed from the combination a motor plan with a set of *kinematic parameters* Θ. Such parameters determine the fine evolution of the resulting trajectories. The transformation from a motor plan to trajectories is denoted with the operator $\odot$, for example:

$$\mathscr{P} = P \odot \Theta \quad .$$

Different kinematic parameters produce different trajectories and consequently different stylisations of the motor plan. This instantiates the previously introduced concept of "style as kinematics". Chapter 4 and Chapter 5 will demonstrate two different trajectory generation methods that result in two different parameterisations of Θ, which take into account a number of principles and methods from the field of computational motor control. These principles will be introduced next in Chapter 3.

Chapter 3

Background

3.1 A Brief History

The inspiration to reproduce hand-drawn or painted artistic styles computationally can be traced back to the infancy of the computer age. The *Algorists* were a group of pioneering artists and computer scientists formed in the 1960 's that employed computers and pen-plotters to generate the first digital algorithmically-based works of art (Leavitt, 1976; Dietrich, 1986). Notably, Frieder Nake (1965) studied the elements of Paul Klee's painting *Hauptweg und Nebenwege* to create his algorithmic work *Hommage à Paul Klee* (Nake, 2005). Michael Noll created computer programs that statistically simulated paintings by Piet Mondrian [1] and Bridget Riley (Noll, 1966; Dietrich, 1986). Since the late 1960's to the end of his days (2016), Harold Cohen developed AARON, an Artificial Intelligence (AI) system capable of automatically generating compositional works of art (Boden, 2003). The program was designed based on Cohen's extensive previous experience as a reputed painter, and has generated thousands of unique artworks, many of which have been exhibited in important galleries across the globe. Inspired by his hand drawing style and study of evolutionary processes, artist William Latham has developed since the late 1980's, together with computer scientist Stephen Todd, a system that evolves organic forms with genetic algorithms (Todd and Latham, 1992).

In the early 1990's, the computer graphics subfield known as *Non Photorealistic Animation and Rendering* (NPAR) emerged with the main goal to produce algorithmic solutions approximating or reproducing artistic techniques or rendering results, with an early focus on animating and rendering painting and drawing styles. For an extensive review of the works in the field refer to (Kyprianidis et al., 2013; Gooch et al., 2010).

With the new millenium, some work started to focus more precisely on understanding and reproducing drawing and painting techniques including via the embodiment of algorithmic models using robots. Deussen et al. (2012) extend painterly techniques from NPAR to the physical world with *e-David*, a repurposed industrial welding robot that is capable of mixing its own colours and paints with a brush and acrylic paint on a canvas. Tresset and Fol Leymarie (2013) have developed the *AIKON II*

[1] Noll (1966) also performed a user study with 100 participants that had to distinguish between computer and human generated versions of the painting. The results showed that 28% of the viewers were able to correctly identify the computer generated version, and 59% preferred the computer generated version.

system, which uses computer vision methods to model the human process involved in portrait sketching production and its embodiment in low cost robotic systems, designed around an articulated arm and wrist as well as an orientable camera-eye. Both projects are on-going and have gone through successive improvements to this day. In our work, we explored the embodiment of some of the methods presented in this thesis using a humanoid robot with compliant control (Berio et al., 2016).

3.2 Beyond painting and drawing: Graffiti production

The scope of this thesis is to develop a deeper understanding of calligraphic production, with a focus on the modern context of graffiti art. This leads us to propose a set of tools that allow the procedural or interactive generation of synthetic graffiti art. We seek to reproduce, as much as possible, the *process* typically used in conceiving and creating graffiti, and also to generate an output that is compatible with robotic and fabrication devices such the ones used by Tresset and Fol Leymarie (2013) and Deussen et al. (2012).

Similarly to painting or drawing, writing graffiti involves knowing how to *move* to produce an artefact, as well as how *perception* impacts the actions involved in the needed gestures. As a result, the remainder of this background chapter covers material ranging from computer graphics methods for curve generation, stylisation and rendering, to principles and mathematical models of human movement, to theories and methods of shape perception and representation. While it is not adequate to thoroughly cover all these subjects in a single chapter of reasonable length, my goal was to include at least the main sources that have informed the development of the methods in this thesis, and to justify some of the choices made.

We start in Section 3.3 with an overview of curve generation and stylisation methods in computer graphics, and then move in Section 3.4 to examine evidence in the arts, psychology and neuroscience, suggesting the importance of bodily movement when producing or appreciating an artwork. We then review in Section 3.5 some principles underlying human movement formation and a number of mathematical models that have been proposed, which prove useful when mimicking traces such as the ones that can be observed in graffiti or calligraphy and form the basis for the methods developed in part I of the thesis. This is followed by Section 3.6, which reviews different methods by which a stylised letterform can be represented and rendered, and Section 3.7, which reviews a number of methods to generate, stylise and segment letterforms. Finally Section 3.8 reviews a topics concerning the computational representation and visual perception of shape, which form the basis for the methods developed in Part II of the thesis.

3.3 Curves in computer graphics

Many modern vector-drawing applications and interfaces provide tools that mimic the appearance of hand drawn strokes and curves. In CAGD applications the two leading approaches to specify curves are: (i) the interactive definition of a *control polygon* defining the shape of a piece-wise parametric curve or (ii) a *sketch-based interface* (Olsen et al., 2009), in which curves are digitised with a device such as a track-pad, mouse or tablet, and then transformed to a piecewise parametric curve. The most widely used curve representations are piecewise *splines*, which are usually defined with either cubic Bézier (Farin, 2002) or B-spline (de Boor, 1978) segments. The choice of a cubic is generally considered to

provide a desirable trade-off between ease of control and smoothness (Foley et al., 1995).

The degree of *parametric continuity* of a curve is denoted with C^n, indicating that a curve parameterisation is continuous up to its nth derivative. For piecewise curves this means that the derivatives up to the nth order are the same where two curve segments join. For example, a C^2 curve has associated continuous tangent vectors and curvature functions, as well as continuous parametric acceleration (change of speed of traversal), and will thus produce a motion that is perceived as smooth. Many CAGD applications are concerned with generating static images, and the requirement of parametric continuity can be relaxed to one of *geometric continuity* (Barsky and DeRose, 1989), denoted as G^n. For example, G^2 continuity implies that connected curve segments share the same curvature at joining endpoints, but may have different parameterisation.

3.3.1 Fairness, beautification and neatness of curves

One specific property of parametric curves that is generally considered desirable in CAGD applications is *fairness*, which relates to continuity in curvature (Levien, 2009a) and implies that a curve is at least G^2 continuous. Farin et al. (1987) propose that a fair curve should possess a curvature profile that is composed of relatively few piecewise linear segments, a property that is analogous to the "french curve" tool used in hand-made drafting. A number of so-called "fairing" methods (McCrae and Singh, 2009, 2011; Levien, 2009a; Baran et al., 2010; Havemann et al., 2013) implement this analogy almost literally, by concatenating curve segments made of *Euler spirals* — curves in which curvature varies linearly with arc-length (Levien, 2009b). The method of Levien (2009a) is specifically aimed at designing outline-based *fonts*, and results in spiral-based splines with desirable properties, such as the ability to generate a perfect circle with four control points. This property, and five others pertaining to the generation of "the most pleasing" interpolating curve for *typography* generation, are enumerated by Knuth (1979) (see Table 3.1), who also shows that all six properties cannot be satisfied at the same time and suggests a spline method developed by Hobby (1986) as an ideal choice. In the same period, font designer Charles Bigelow, a close collaborator of Knuth, emphasises the importance of *hand movements* when considering letter design. Bigelow also notes, in an interview (Wang, 2013), that Bézier curves, and splines in general "are usually pleasant, but they are more limited than the shapes that result from the living hand moving a traditional tool through a complex path".

Property	Short Description
Invariance	to scale, rotation and translation.
Symmetry	invariance to cyclic permutation or reversal of order.
Extensionality	such that adding a control point on the curve should not modify it.
Locality	such that the geometry of a curve segment defined between two control points only depends on those two points and directly adjacent ones.
Smoothness	such that the curve should be sufficiently differentiable.
Roundedness	such that given four equidistant control points on a circle, the curve will define the same circle.

Table 3.1: The six properties of the "most pleasing curve" according to Knuth (1979).

Similar, and sometimes equivalent, to the concept of curve fairing, *curve beautification* or *neatening* is the process of inferring geometric or structural constraints, primitives or graphic intentions from user free-hand input (Igarashi et al., 2007). Zitnick (2013) beautifies handwriting and sketches by averaging parts (tokens) of the input with previous specimens by the same user. The averaging process smooths out imperfections while maintaining consistency with the user's style. Thiel et al. (2011) interactively neaten traces made with a pointing device by analysing the velocity of the movement. The system smooths out the input at a degree proportional to its velocity, on the basis of the observation that users commonly slow down their movements when they intend to create a more precise drawing.

One limitation of faring and beautification or neatness methods, is that the output usually consists of many curve segments the location of which depends on the input geometry and can thus be difficult to edit. The same drawback generally holds also for Bézier curves with manually defined control points, which must be placed at locations that depend on the desired curve geometry and smoothness but do not necessarily reflect any perceptually salient feature along its trace (Yan et al., 2017; Levien and Séquin, 2009). This can be especially challenging when attempting to mimic curves such as the ones that can be seen in handwriting and calligraphy"‘ (Wang, 2013). Yan et al. (2017) address this problem in the context of interactive Bézier curve editing, with an interface in which control points are constrained to coincide with absolute curvature maxima along the generated curve. The method stores an underlying representation made of quadratic curve segments, which is then converted to cubic curves in real time.

In this thesis we propose a different and hybrid approach that is especially aimed at reproducing calligraphic traces. We automatically generate trajectories that are geometrically *and kinematically* similar to the ones that could be made with a skilled movement and a sketch-based interface. However these trajectories can be defined and edited with an interface using control points similar to those found in traditional curve generation methods.

3.3.2 Curve stylisation

A number of methods are aimed at generating stylised curves, often relying on a data-driven approach. For example Lu et al. (2012) stylise digitised traces by adaptively concatenating segments from examples made by expert artists on tablets with a high number of degrees of freedom, such as based on pen tilt and pressure. The input trace acts as a "guiding curve" for the inference of the missing pen-tilt and pressure information which is matched from the available examples with dynamic programming, thus allowing the creation of more realistic brush renderings. Li et al. (2013) assume that the "style" of a 2D outline is represented by high frequency decorative features rather than structural or topological ones, and build a system that can either de-emphasise or exaggerate stylistic features, or transfer and blend these features between outlines. Hertzmann et al. (2002) propose a data-driven approach to stylise an input curve based on a database of examples. The problem is expressed as follows:

Definition 3.3.1 (Hertzmann et al. (2002))**.** Given an input curve A, its stylised version A' and a second curve B, learn a mapping between A and A' in order to generate a stylised version B' of B, such that the analogy $A : A' :: B : B'$ holds.

The proposed implementation is functional but requires the tuning of a large number of parameters, and does not run in real time. Lang and Alexa (2015) approach the same problem from a probabilistic

standpoint and use a Double Hidden Markov Model to achieve a comparable result in real time and with very few parameters to tune. Freeman et al. (2003) implement a similar style transfer system by employing K-nearest neighbors (KNN) with locally weighted linear regression (Stulp and Sigaud, 2015). The system stylises an input drawing given a large number of samples of stylized lines. In Chapter 9 we adopt an approach similar to the one of Hertzmann et al. (2002) and Li et al. (2013), but consider the kinematics of a movement as a descriptor of style.

A few examples exist that have exploited movement kinematics or dynamics to generate stylised curves. The approach of Lu et al. (2012) can be categorised as one such example, since the resulting stylisation depends on the skilled movements of expert artists. Haeberli (1989) implemented DynaDraw, a computer program that allows the user to interactive generate strokes evocative of calligraphy by simulating a mass-spring system attached to the mouse position. Levin et al. (2013) generate abstract alphabets with a genetic algorithm combined with a physics simulation of a hand and pen. House and Singh (2007) generate sketch based renderings, by using a Proportional Integral Derivative (PID) controller to define the trajectory of a pen that follows the connected contours of a 3D mesh. Thompson (2010) generates calligraphic effects on letterforms by optimising the evolution of a point mass along a series of user defined spatial constraints. AlMeraj et al. (2009), use a well known model of reaching movements that minimises jerk (i.e. changes in acceleration) (Flash and Hogan, 1985) to mimic the visual qualities of hand drawn lines passing trough point triplets. Fujioka et al. (2006) adapt a similar optimal control model to optimize the location of B-spline control points (Egerstedt and Martin, 2009), in order to generate Japanese calligraphy and to smoothly vary brush width.

3.4 Movement perception and representation

The hand-drawn curves that can be observed in art forms such as graffiti (Cooper and Chalfant, 1984) and calligraphy (Mediavilla et al., 1996) are usually, if not always, the result of skillful and expressive movements that require years to master. To put it in the (skillfully written) words of calligrapher Karen Knorr (Briem et al., 1983): [2]

> The construction of the knot isn't what counts. What you must learn is the *movement*. Three hundred knots later...

This suggests the hypothesis that the perceived visual quality of a static calligraphic trace, depends (at least in a significant way) on the properties of a motion that have generated it. If this is the case, having parametric control of these properties when generating a synthetic motion should produce a perceived visual quality similar to traces produced by skilled humans. In the following sections we will examine some evidence in favour of this hypothesis, first in the art-history literature and then in the domains of psychology and neuroscience.

3.4.1 Movement in the arts

It proves difficult to find an account of the motor act of drawing or painting in the Western art-historical literature, which is often focused on the methods, techniques and compositional or formal aspects

[2] Quote from a special issue of the Visible Language journal, entirely written by hand by a number (51) of calligraphers, who answered with drawings and writing the question: "What parts of your work give you the most trouble?". It can be accessed online at `http://visiblelanguagejournal.com/issue/65`.

of art works and styles (Fong, 2003; Seeley, 2013). The same can be said for a large part of works in Computer Graphics within the NPAR sub-field (Kyprianidis et al., 2013), with the exception of a few methods that we have previously mentioned in Section 3.3.2. However, in his well-regarded treaty "Art and Illusion", Gombrich (1977) notes that:

> The word *style* of course, is derived from "stilus", the writing instrument of the Romans, who would speak of an "accomplished style" much as later generations spoke of a "fluent pen".

Rosand (2002) points out the importance of the artist's movements in works such as the ones by Leonardo, and considers drawing as an act of "self projection" of body actions onto the resulting traces. To emphasise this Rosand also writes (Rosand, 2013)

> The gesture of drawing is, in essence, a projection of the body, and, especially when viewing the drawing of the human figure, we are inevitably reminded of that.

And

> Responding to drawings, we make our way, through line, to the originary impulse of the draughtsman. Interpretation involves a connecting act of re-creation, the self-projection of the viewer re-imagining the process of drawing.

Movement is fundamental at the conceptual level for modern artists such as Paul Klee and Cy Twombly. In his notebooks, Klee (1961) famously defines a line in a drawing as "A point that has gone for a walk". Twombly's work can resemble graffiti (Kaushik, 2013) and is made with spontaneous gestures "made for their own experience" and evoking the same experience in the beholder (Rosand, 2013).

While references to generative movements are rather scarse in the Western art-historical literature, the situation is different with East-Asian art history, where calligraphy is recognised for centuries as one of the most important art forms. Art historian Wen C. Fong (2003) observes that:

> Rather than color or light, the key to Chinese paintings lies in its calligraphic line, which bears the presence or physical "trace" (*ji*) of its maker.

The author notes how terms commonly used to describe Chinese calligraphy and painting, such as *biji* ("trace of the brush"), *moji* ("trace of ink"), *yi* ("made with spontaneity and naturalness"), *qiyun shengdong* ("breath resonance" or "life motion"), are examples of how the static art works are a record of the artist's skilled movements and convey their physical presence. Similar principles hold for the Japanese art of calligraphy (Ferri, 2016), which is called *shodo*, a term that literally means "the *way* of writing" (Albertazzi et al., 2015)

3.4.2 Perception of movement in static forms

Common artistic knowledge and intuitions, as well as art theories such as the ones discussed above by Rosand (2002) and Fong (2003), suggest that viewing a static work of art can evoke the gestures used by an artist to create it. This hypothesis is further grounded in psychological and neuro-scientific evidence suggesting that indeed the visual perception of marks made by a drawing hand triggers activity in the *motor areas* of the brain (Freedberg and Gallese, 2007; Longcamp et al., 2003). This further induces an approximate mental recovery of the (likely) movements and gestures underlying the artistic production

(Freyd, 1983; Pignocchi, 2010), and such recovery influences its aesthetic appreciation (Leder et al., 2012).

Freyd (1987) proposes that handwriting recognition is done by infering the motion used in its production. The author posits that time is an intrinsic part of representation and static representations are just special cases of dynamic representations. Pignocchi (2010) proposes that drawings are planned and perceived at the level of Atomic Graphic Schemes (AGS), visuomotor primitives that associate the visual aspects of a trace with a motor primitives and are automatically triggered when producing or perceiving a drawing. AGS are combined in the form of Molecular Graphic Schemes, that with experience and practice are assimilated as a single AGS, thus influencing the perception, production and planning of a drawing. The validity of this model would explain the way in which experience in drawing influences the perception of drawing.

It has been discovered that macaque monkeys possess a set of "mirror neurons" in the pre-motor cortex, which are activated both when *producing* as well as when *observing* a movement (Gallese et al., 1996). Brain imaging data suggests a very high likelihood that a similar mechanism is present also in humans (for a review on the subject refer to Oztop et al. 2013). Freedberg and Gallese (2007) propose that a similar mechanism may be activated in the brain during the perception of art works. Figurative art works that depict actions or movement produce a sense of bodily "empathy" in the viewer. The traces left by an artist on a canvas induce in the viewer a form of mental simulation that recovers the artist's gestures and intentions (Freedberg and Gallese, 2007). In an EEG study Umilta et al. (2012) detected clear activity in the cortical motor system of subjects when viewing the cuts on a canvas performed by artist Lucio Fontana, even when the observers were not familiar with the artist's work. Leder et al. (2012) observe a positive correlation between the aesthetic appreciation of a painting with the execution of movements that are similar to the ones used to produce the painting.

James and Gauthier (2006) and later Longcamp et al. (2009) have performed fMRI experiments to find the interaction between visual and motor perception during letter writing and visualisation. The authors show that writing letters activates areas of the brain associated with visual perception, and visualising letters activates areas of the brain associated with motor control. The authors propose that these areas of the brain form a "letter network" (Longcamp et al., 2009) that is activated both during perception and production of letterforms. In a series of MEG, fMRI and EEG studies (Longcamp et al., 2006, 2011; Wamain et al., 2012), it is shown that handwritten letters produce a higher motor cortex activation than printed letters. The execution of a motor task during the observation suppresses these activations (Wamain et al., 2009). Handwritten letters also trigger the activation of an area of the brain (SMA) which is commonly attributed to the planning of complex sequential movements and is also employed to inhibit movements produced by the motor cortex (Longcamp et al., 2011). While the activation of areas of the brain involved in motor control is higher for handwritten letters, activity is present also for printed letters (Longcamp et al., 2011). This suggests that letters in general trigger some form of mental simulation of motor action.

In summary, these results and observations suggest the important role of movement, even when observing the static traces of a drawing or painting, and in particular those that can be seen in written art forms such as calligraphy. To find additional evidence for this hypothesis and to take it into account when generating synthetic traces, it is thus useful to study more in depth some principles underlying

human movement formation and some relevant computational models.

3.5 Motor control

The scientific study of human and biological movement is a highly interdisciplinary field of research, which has evolved at the intersection of psychology, neuroscience, mathematics, physiology, robotics and human computer interaction. In our study we focus on upper limbs in humans and their simulation in robotics, i.e. the movements of the arm-wrist-hand system, which we refer to as the "human arm" for simplicity. These movements can be described in 3 types of coordinate systems (Mussa-Ivaldi et al., 2004):

- *Endpoint coordinates*: positions (and orientation) of the hand or end effector.
- *Actuator coordinates*: human muscle activations or robot arm motor torques.
- *Generalised coordinates*: joint angles between each limb segment.

Inverse kinematics is the process of finding the transformation between endpoint coordinates and generalised coordinates. *Inverse Dynamics* is the process of transforming generalised coordinates into actuator coordinates, i.e. computing the muscle activations, or actuator torques for moving the limb given a set of joint angles. During the execution of a simple line with a brush, or a reaching motion of the hand, the central nervous system (CNS) is faced with inverse kinematics and dynamics problems that are ill-posed (Schomaker, 1991; Mussa-Ivaldi et al., 2004), i.e. there may be either no single solution or an infinite number of these.

In the context of generalised coordinates, the human arm is typically modelled with seven Degrees of Freedom (DOFs) which already implies an infinite number of possible arm configurations for a given end position of the hand. Furthermore, each joint of the arm is controlled by a number of muscles, which in turn are composed by hundreds of smaller components known as motor-units. This raises the effective number of DOFs of the human arm to the order of thousands (Turvey et al., 1982). The problem of motor coordination given the "abundance of degrees of freedom" — or redundancy problem – was noted by Nicolai Bernstein in his pioneering studies on human motion (Bernstein, 1967). Bernstein, proposed the concept of *synergies* as a mean for organisms to overcome the redundancy problem. Synergies are interactions between muscles and joints that (in principle) constrain the possibilities of motion and simplify the motor control problem. It has been shown that the physical properties of the body can greatly simplify the motor control problem, an example can be seen with swinging of legs during gait (Rosenbaum, 2009), simulated in robotics with "passive dynamic walkers" which are able to walk down slopes thanks to their mechanical constraints (Collins et al., 2005).

3.5.1 Principles and invariants

Through several years of experimental research on human movement, a series of "invariants" have emerged that seem to relate the dynamic (time, speed) and figural (curvature, shape) aspects of human hand motions (refer to table 3.2 for a summary). While such principles are not set in stone, they constitute a general basis for the advancement of new theories, and a benchmark for the validation of mathematical models that have been created. For an extensive review on the subject refer to the book by Rosenbaum (2009). Below we give some details on the invariants and other principles most relevant to our thesis.

Principle	Short Description	References
Speed accuracy tradeoff	Movement time is inversely proportional to required accuracy.	(Woodworth, 1899; Fitts, 1954; MacKenzie and Buxton, 1992; Plamondon and Alimi, 1997)
Isochrony	Movement time is approximately independent of movement extent.	(Freeman, 1914; Viviani and McCollum, 1983; Thomassen and Teulings, 1985; Jordan and Wolpert, 1999)
Isogony	Tangential velocity is proportional to radius of curvature	(Viviani and Terzuolo, 1982; Thomassen and Teulings, 1985)
Power laws (2/3, 1/3)	Velocity and curvature radius are related by a power law	(Lacquaniti et al., 1983; Viviani and Schneider, 1991; Plamondon and Guerfali, 1998a; Flash and Handzel, 2007)
Open loop movements	Movements can be executed in an open loop manner, and do not necessarily require continuous visual/proprioceptive feedback.	(Taub and Berman, 1968; Polit and Bizzi, 1978; van Doorn and Keuss, 1993; Schomaker, 1991; Jordan and Wolpert, 1999)
Equi-affine geometry	Movement segments have constant velocity in equi-affine space.	(Flash and Handzel, 2007; Polyakov et al., 2009)
Motor primitves	Complex movements are planned and executed at the level of a discrete number of simpler units of action	(Lacquaniti et al., 1983; Teulings and Schomaker, 1993; Morasso, 1986; Polyakov et al., 2009; Morasso and Mussa Ivaldi, 1982; Rosenbaum et al., 1995; Mussa-Ivaldi and Bizzi, 2000; Flash and Hochner, 2005)
Bell-shape	The speed profile of rapid and straight reaching motions can be described by a “bell shaped” function.	(Morasso, 1981; Nagasaki, 1989; Abend et al., 1982; Plamondon, 1995)

Table 3.2: Summary of principles that have been observed in human upper limb movements.

3.5.1.1 Motor equivalence

Humans are able to perform the same type of movement regardless of the sets of muscles and limb used. For example, upon request, a person will usually be able to trace a figure eight with a finger, the hand, an elbow, a foot or even with the tip of the nose. This is known as the principle of *motor equivalence*. On the basis of this principle, Bernstein (1967) hypothesises that the CNS represents movement in endpoint coordinates. Morasso (1981) supports this hypothesis experimentally, and shows that in

reaching arm motions at random targets the joint angular velocity profiles vary greatly within trials, while the tangential velocity profiles are always characterised by a bell-shape with a single peak.

3.5.1.2 Isochrony.

Since early in the 20th century (Freeman, 1914), various experiments have demonstrated the tendency of humans to keep the time of movements relatively independent across different size ranges. In other words, velocity increases proportionally to movement extent (Denier and Thuring, 1965; Ghez et al., 1997). This principle is commonly referred to as *isochrony* (Viviani and McCollum, 1983), where *global* isochrony refers to movements and trajectories as a whole, and *local* isochrony refers to parts of a movement (Jordan and Wolpert, 1999). In a study of handwriting in adults and children, Freeman (1914) observed that, especially in adults, parts of a letter with a different extent were executed in approximately equal times. Thomassen and Teulings (1985) hypothesise that isochrony is caused by the narrow frequency band of the output of the motor system and observe in a study of handwriting that past a letter size threshold of 16 cm, the isochrony principle breaks down, and movement time increases as a power (1/2) function of size.

3.5.1.3 Isogony and Power laws

Already by the end of the 19th century, Jack (1894) noticed that "the curved parts of letters and figures are more slowly formed than the rectilinear parts, and that the velocity of a curve varies, roughly speaking, with the radius of curvature". This phenomenon was also observed by Freeman (1914) in his study of handwriting movements, where he observed that sharper turns on a curve corresponded with a decrease in speed. Viviani and Terzuolo (1982) formalise this invariant with the principle of *isogony*, stating that a trajectory goes through "equal angles in equal times" with a tangential velocity that is proportional to the radius of curvature, weighted by a constant K that varies in a piecewise manner across the movement. This piecewise variation of K gives an indication that motion can be segmented in sub-movements, or units of action that define the velocity profile of the motion. Lacquaniti et al. (1983) observe that for certain families of movements, angular velocity and curvature are related by a *two-thirds power law*,[3] which in terms of tangential velocity gives:

$$\|\dot{\boldsymbol{x}}\| = Kr^{1/3} \quad , \tag{3.1}$$

where $r = r(t) = 1/\kappa$ is the radius of curvature.

The power law has been shown to be valid only for certain classes of movement. For example, Plamondon and Guerfali (1998a) show through an analysis by synthesis experiment that for complex handwriting movements the power law holds only for segments of the trajectory that are hyperbolic or elliptical. Nevertheless, the general principle seem to hold that (i) steeper turns are executed at a lower speed and that (ii) curvature and velocity are related by a power law. This gives a useful hint for the intrinsic relationships between dynamics and geometry in hand drawn movements. More recently, a series of studies (Flash and Handzel, 2007; Polyakov et al., 2009; Maoz et al., 2009) has proposed that the perception and the production of movement has an underlying structure that is non-Euclidean. Flash and Handzel (2007) observe that movement invariants such as the power law or isochrony can be

[3] For the angular velocity ω and curvature κ the power law is: $\omega = K\kappa^{2/3}$, which in terms of tangential velocity becomes: $\|\dot{\boldsymbol{x}}\| = K\kappa^{-1/3}$.

interpreted geometrically by analysing movement through the lens of affine and equi-affine geometries. Indeed, given the Euclidean radius of curvature $r(t)$ and arc length s, the equi-affine arc length σ_1 is defined by:

$$\frac{d\sigma_1}{ds} = r(t)^{\frac{1}{3}} \quad , \tag{3.2}$$

which corresponds to the above power law.

3.5.1.4 Feedback

In a seminal study of human movement, Woodworth (1899) studied the variability of rapid aiming tasks by analysing pen movements constrained to a slit of varying width. The study resulted in the following observations:

- Aiming motions can be subdivided in two phases: an initial *ballistic* phase, and a subsequent *corrective* phase.
- Higher movement speed resulted in a higher variability of the motions and error in reaching the targets.
- With no visual feedback, the movements showed a higher variability, which was approximately constant regardless of speed; however, past a certain movement speed, the vision and no-vision conditions converged to a similar error and variability.

This last observation led Woodworth to hypothesise that visual feedback has a limited rate (250 Hz), which limits its applicability to very rapid movements and suggests that movements can be made in a completely open-loop manner. This has been confirmed more recently, with the rate of visual feedback being between 100 and 200 Hz (Rosenbaum, 2009, p.228). In a study about well learned movements, van Doorn and Keuss (1993) showed that the variability of handwriting parameters with or without vision becomes less significant.

3.5.1.5 Variability

Following Woodworth's line of work, Fitts (1954) employs an information-theoretical approach to study rapid aiming and reaching movements. The results demonstrate a "speed-accuracy tradeoff", i.e. the tendency of movement time to increase with distance and to decrease with a reduction of required accuracy. Fitts explained these results with a fixed information-transmission capacity of the motor system and quantified the relation between time, precision and distance with an equation commonly referred to as "Fitts' Law" (Fitts and Posner, 1967). Numerous other extensions of Fitts' law have been developed through the years in order to improve its predictive accuracy; for a review the reader is referred to Plamondon and Alimi (1997).

Harris and Wolpert (1998) suggest that neural signals are noisy, and that noise is proportional to the amplitude of the neural signal. This observation results in a *minimum-variance model* of point-to-point movements, where movement velocity and time result from the minimisation of a trade-off between end-point accuracy and movement speed. Todorov and Jordan (2002b) suggest that the redundancy of the musculoskeletal system acts as a "noise buffer" or "uncontrolled manifold" (Scholz and Schöner, 1999), which is taken advantage of in order to "project" the errors caused by sensory delay and neuro-motor noise onto dimensions that minimise the effect on the motor task. This results in

a *minimal intervention principle* of trajectory formation (Todorov and Jordan, 2002a), where deviations from an average and maximally smooth trajectory are corrected only if the required precision is high.

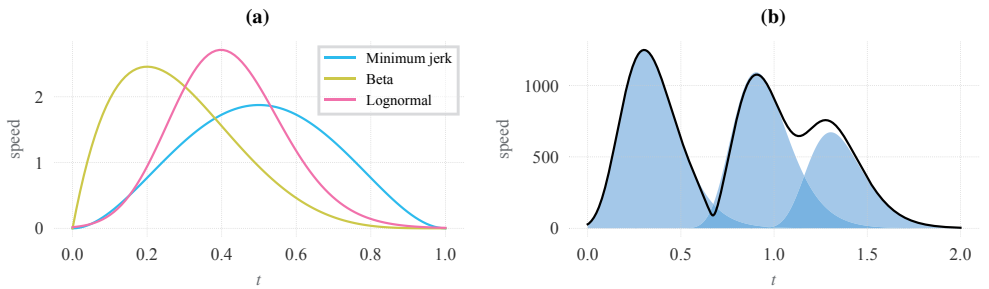


Figure 3.1: Bell shaped speed profiles. **(a)** Examples of functions used to describe the characteristic bell shape: minimum jerk, a Beta function and a lognormal. **(b)** Superposition of lognormals and the resulting speed profile.

3.5.1.6 Kinematics

The tangential velocity profile of point-to-point aiming movements typically assumes a "bell shape" (Morasso, 1981; Flash and Hochner, 2005; Plamondon et al., 2014), variably asymmetric depending on the rapidity of the movement (Nagasaki, 1989; Plamondon et al., 2013). The bell shape has been modeled with a variety of techniques (Figure 3.1(a)), which include sinusoidal functions (Morasso and Mussa Ivaldi, 1982; Maarse, 1987; Rosenbaum et al., 1995), Beta functions (Lee and Cho, 1998; Bezine et al., 2004), optimisation methods (Flash and Hogan, 1985; Hoff, 1994), and lognormals (Plamondon, 1995). Plamondon et al. (1993) and later Rohrer and Hogan (2003) have shown that between different fitting curves the best velocity profile of point movements is given by a support-bound lognormal, which is variably asymmetric (Figure 3.1(b)).

Hand movements are typically smooth and result in trajectories that can be explained with the minimisation of the square magnitude of high derivatives of position with distinctive names such as "jerk" for 3rd order (Flash and Hogan, 1985), "snap" for 4th order (Flash, 1983) and "crackle" for 5th order (Dingwell et al., 2004). The velocity of smooth hand movements can be reconstructed with the superposition of a discrete number of target-directed "ballistic" sub-movements that are also characterised by the stereotypical bell-shaped velocity profile (Rohrer and Hogan, 2006; Flash and Henis, 1991; Leiva et al., 2017).[4] Different sub-movement durations or activation-times produce different but yet smooth kinematics (Flash and Henis, 1991).

The velocity peak of each sub-movement produces a peak in the superimposed speed profile and a consequent minimum between consecutive peaks (Figure 3.1(b)). Consistent with the isogony principle, this also results in an (absolute) maximum of curvature. Consequently, curvature maxima are

[4]These sub-movements are often referred to as "strokes" (Teulings and Schomaker, 1993; Morasso, 1986; Sosnik et al., 2004). However, in the wider context relevant to this thesis, the term "stroke" is also commonly used to refer to the mark resulting from a complex movement made with a writing tool in contact with a surface. We will adopt the latter interpretation of the term, and refer to movement-units as "movement primitives" or "sub-movements".

generally considered good indicators for the segmentation of a movement into basic units (Brault and Plamondon, 1993a; Meirovitch and Flash, 2013; De Stefano et al., 2005). With experience, a movement tends to become smoother (Sosnik et al., 2004; Rohrer and Hogan, 2003; Plamondon et al., 2013) and the number of velocity peaks decreases. This phenomenon is known as *co-articulation* and can be interpreted as the *chunking* (or fusion) of movement primitives at the planning level (Sosnik et al., 2004).

3.5.1.7 Motor primitives and representation

The form in which the human brain represents movement is still an open question. For example there is a long standing debate between a cognitive or "computational view" of motor planning and representation and "dynamical-system view" (Rosenbaum et al., 2007). The computational view defends the existence of some form of abstract representation of movement often referred to as a *motor program*, *engram* or *schema* (Keele and Summers, 1976; Schmidt, 1975; Rosenbaum et al., 2007). An early proponent of this view is Lashley (1951), who describes complex and skilled motions as the combination of "units of action" that are centrally combined to form an hierarchically organised and context dependent plan. The decomposition of a movement into ballistic sub-movements is a modern example of such an approach. The dynamical-system view suggests that perception and action emerge from the continuous interaction of mental processes, muscle/limb dynamics and the surrounding environment (Kelso and Saltzman, 1982; Turvey et al., 1982; Newell and Vaillancourt, 2001). Organised movements are achieved through "coordinative structures": groups of muscles that are employed cooperatively in the resolution of a task (Newell and Vaillancourt, 2001), or as previously proposed by Bernstein 1967: *synergies*. In line with this view, Schaal et al. (2007) propose Dynamic Movement Primitives (DMPs), which are popular in robotics and reproduce both discrete and oscillatory motions by variably modulating a mass-spring-damper system with a forcing term.

While the computational and dynamical-system views are often put in contrast, they are not necessary mutually exclusive, but rather can be seen as complementary descriptions of the complex process of movement planning and formation (Woch and Plamondon, 2003; Krampe et al., 2002). The general consensus is that there exists some form of mental and neural representations of movement that guides the motions of the human body (Flash and Hochner, 2005). The ability of humans and animals to perform movements without sensory feedback (Schomaker, 1991; Bizzi and Polit, 1979), the observation of invariants in hand motions (Lacquaniti et al., 1983; Viviani and McCollum, 1983; Viviani and Terzuolo, 1982), the ability of humans to mentally visualise a motion (Jeannerod, 1995), increasing reaction times with increasing movement complexity (Henry and Rogers, 1960), are indicative of the existence of a central representation of movement in some form.

Flash and Hochner (2005) propose that complex motions (in humans, but also in animals including invertebrates) can be broken down into elementary building blocks, i.e. motor primitives, that are combined and superimposed to produce complex motion. The way in which a movement can be represented depends on the underlying model. For example motor primitives consisting of ballistic submovements, intrinsically define a *motor plan* consisting of a sequence of consecutive aiming targets. These are also known as "virtual targets" (Djioua and Plamondon, 2009), because they can be seen as imaginary locations at which a sub-movement is aimed. Other models that operate in endpoint coordinates describe a movement with a series of "via-points" (e.g. Edelman and Flash 1987 or Wada and Kawato 1995) that, unlike virtual targets, are located along a trajectory. Models that operate in gener-

alised coordinates can represent a movement as sequences of poses, while ones that operate in actuator coordinates describe movement with equilibrium trajectories (Bizzi et al., 1992; Feldman, 1966) or force fields (Mussa-Ivaldi and Bizzi, 2000) that determine muscle coactivations. Table 3.3 gives a summary view of the main representations found in the literature.

Representation	Short Description	Coordinates	References
Via-points	Landmark points along the movement trajectory.	endpoint,generalised	(Flash and Hogan, 1985; Viviani and Flash, 1995; Flash, 1983; Todorov and Jordan, 1998; Hoff, 1994; Uno et al., 1989; Harris and Wolpert, 1998; Edelman and Flash, 1987; Bullock et al., 1993; Grossberg and Paine, 2000; Meulenbroek et al., 1996)
Virtual targets	Imaginary loci at which ballistic sub-movements are aimed.	endpoint	(Morasso and Mussa Ivaldi, 1982; Maarse, 1987; Plamondon and Guerfali, 1998b; Plamondon et al., 2009; Bezine et al., 2004)
Poses	Configurations of the arm at discrete times during a movement.	generalised	(Rosenbaum et al., 1995; Meulenbroek et al., 1996)
Attractors	Attractors for a dynamical system.	endpoint, generalised	(Del Vecchio et al., 2003; Schaal, 2006)
Force fields	Superposition of force fields that describe muscle activation.	actuator	(Bizzi et al., 1991; Mussa-Ivaldi, 1997; Mussa-Ivaldi and Bizzi, 2000)
Oscillatory	Oscillation amplitude and phase modulation.	endpoint	(van der Gon et al., 1962; Vredenbregt and Koster, 1971; Dooijes, 1983; Hollerbach, 1981)

Table 3.3: Summary of movement representations.

In the context of this thesis, we are interested in reproducing trajectories with properties that are consistent with those observed in human hand and arm movements. At the same time, we seek a representation that allows for a simple user interface, similar to the control polygon typically used for parametric curve design in CAGD. While the way in which movements are planned remains an open question, motor equivalence (Bernstein, 1967; Morasso, 1981) and recent experiments (Torres et al., 2013; Wong et al., 2016) indicate that human hand movements are likely planned at the level of endpoint coordinates. Intuitively, specifying movements at the joint level or muscle level would be quite impractical for our application. However taking posture, or even some model of muscles into account is definitely an interesting area of future research. As a result, we will principally focus on computational models and representations that operate at the level of endpoint coordinates and on movement representations consisting of either virtual targets or via-points.

3.5.2 Trajectory formation

We consider in this section two main strategies based on: (i) optimal control, and (ii) ballistic submovements.

3.5.2.1 Optimal control

A series of models propose that trajectory formation can be explained through an optimisation that minimises some form of objective function or “cost” (Engelbrecht, 2001). Based on the observation

that hand motions are intrinsically smooth, Flash and Hogan (1985) propose the *minimum-jerk* model (Hogan, 1982), in which hand movements are planned in order to minimise the squared magnitude of jerk (the 3rd order derivative of position). Flash and Hogan (1985) derive the optimal solution in terms of a quintic polynomial, describing the evolution of straight "point-to-point" movements, or a curved movement that interpolates a so-called "via-point".[5] The minimum jerk model also predicts the time occurrence of one or more via-points. For trajectories with one single via-point the time occurrence is the solution to a 9^{th} degree polynomial of the total movement duration. Todorov and Jordan (1998) develop a *constrained minimum-jerk model* computing the kinematics of a minimum jerk motion given a predefined path. The model predicts the time occurrence of multiple via-points that minimise jerk with a non-linear optimisation procedure. Viviani and Flash (1995) show how complex trajectories can be achieved by specifying velocity and acceleration constraints in correspondence with multiple via-points with experimentally determined passage times, producing trajectories that are consistent with the two-thirds power law (Eqn. 3.1).

Flash (1983) also derives the *minimum-snap* model, which minimises the square magnitude of snap (4^{th} derivative of position), resulting in a solution to a polynomial of degree 7. Flash shows experimentally that the *minimum snap* model gives an overall better approximation for the tangential velocity profiles of point-to-point movements, while the *minimum jerk* model gives better results for curved movements. Other minimisation based methods include the *minimum time* model (Hoff, 1994), which adds a time penalty to the minimum jerk cost function, the *minimum torque* model (Uno et al., 1989), which also takes the dynamics of the musculoskeletal system into account, and the *minimum variance* model (Harris and Wolpert, 1998), which minimises end point variance. The latter model also predicts asymmetric velocity profiles, as opposed to the other methods that predict perfectly symmetric velocity profiles (Tanaka et al., 2004). Egerstedt and Martin (2009) show the equivalence between several forms of splines and control theoretic formulations of linear dynamical systems. The authors show that polynomial splines of degree $2n-1$ correspond to the output of a controller that minimises the squared magnitude of the nth order derivative of position.

Consistent with the *minimal intervention principle*, Todorov (2004) and Wolpert et al. (2011) propose *optimal feedback control* as a model of trajectory formation, in which the biological controller uses predictive control and internal dynamic models to infer the outcomes of a movement and to overcome the inherent noisiness and delays of the sensimotor system. The objective is not to minimise a cost function such as based on jerk, snap, torque or variance, but rather it is the optimisation of a controller performance depending on the task constraints. A practical implementation of these methods based on machine learning and model predictive control has been proposed by Calinon (2016a) in the context of programming by demonstration in robotics. We will demonstrate an extension of this method for the interactive generation of calligraphic trajectories in Chapter 5 of this thesis.

3.5.2.2 Ballistic sub-movements

The previous group of control optimisation methods result in trajectories that possess regularities that are experimentally consistent with the ones observed in human movements. Another important set of methods used to describe a movement is through the combination of one or more ballistic sub-

[5] A via-point is an intermediary passage point that is interpolated by a minimum jerk trajectory. It plays a role equivalent to the position of a spline knot.

movements, characterized by a bell-shaped velocity profile.

An early example of such a model, is developed by Morasso and Mussa Ivaldi (1982), who describe complex handwriting movements with the superposition of ballistic sub-movements characterised by a sinusoidal velocity profile and a trajectory trace given by a B-spline. Another early example is by Flash and Henis (1991) who generate complex motions by superimposing multiple target directed sub-movements described with the minimum jerk model.

One particularly important class of ballistic models of trajectory formation is developed by Plamondon et al. in what is known as the "kinematic theory of rapid human movements" (or kinematic theory for short) (Plamondon, 1995). This theory assumes that movement results from the parallel and hierarchical interaction of a large number of neuromuscular units that are modeled as linear subsystems. With this premise, Plamondon et al. (2003) use the Central Limit Theorem to prove that the impulse response of the system as a whole to a centrally generated command asymptotically converges to a *lognormal*, resulting in a varying asymmetric velocity profile. The kinematic theory includes a number of models that describe movements of varying complexity. The *Delta Lognormal* model (Plamondon, 1995), describes the velocity of rapid point-to-point movements with the synergy of an agonist component and an antagonist component acting in opposite directions. The *Vectorial Delta lognormal* model (Plamondon and Guerfali, 1998b) and the *Sigma Lognormal* model (Plamondon et al., 2009) describe complex hand trajectories via the vectorial superposition in time of movement primitives each with a lognormal speed profile.

Plamondon et al. (2013) show that with increasing experience, the rapid movements made by an adult when writing converge towards "lognormality". In other words, with practice towards expertise, the velocity of the trajectories can be more and more precisely described by a sum of lognormal curves. In Chapter 4 we will extend the Sigma Lognormal model, showing how it can also be used as a powerful and interactive curve and trajectory generation tool.

3.5.3 Graphonomics: Models of drawing and handwriting movement

The first "computational" (and embodied) model of handwriting can perhaps be attributed to Jacquet Droz (1721-1790), the inventor of the wristwatch (Rosheim, 1994). Droz created a series of humanoid automata, the most complex being the "writer" (made of 6,000 mechanical pieces) that can write script powered by a clock-work like engine. The machine is programmable, in the sense that the combinations of letters can be set by switching modular letter elements — serving as physical records of the needed movements — in the mechanism.[6]

Most of the more recent research on handwriting synthesis and analysis has been conducted under the field known as *graphonomics*,[7] the scientific field "concerned with the systematic relationships involved in the generation and analysis of the handwriting and drawing movements, and the resulting traces of writing and drawing instruments" (Kao et al., 1986).

[6] Three of the original automata — the musician, the draughtsman and the writer — are still functioning and maintained at the Museum of Art and History of Neuchatel, Switzerland.

[7] Not to be confused with the less scientifically rigorous field of *graphology*, which studies the correlation between properties of handwriting and personality traits.

3.5.3.1 Principles

Handwriting movements are characterised by the previously summarised principles and invariants (§ 3.5.1) together with a number of additional properties that have emerged with studies in the field of graphonomics. For example, both Dooijes (1983) and Maarse (1987) observe that handwriting is well described in the plane with an *oblique* coordinate system, where the origin can shift from left to right due to the forearm motion, and the oblique axes span the space covered by the combined movement of the two forefingers and the thumb and slight rotational movements of the wrist. The angle between the two axes is shown to vary between 30° for drawing motions and 90° for handwriting. We will use a similar principle to vary trajectory stylisations in Chapter 5.

Consistent with the hypothesis of chunking at the motor level, Teulings et al. (1986) note that with well-practiced handwriting gestures, planning occurs at the level of more complex combinations of movement primitives. Edelman and Flash (1987) identify four basic types of curved primitives that can be combined to generate cursive handwriting. The authors show that the minimum-jerk model with a single via-point is not sufficient to capture all types of strokes, unless a velocity constraint is specified at the via-point. On the other hand, a minimum-snap trajectory (Flash, 1983) with one via-point is capable of describing these shapes, and the constraints can also be defined geometrically by specifying the desired slope of the trajectory at a via-point. As a result, the dynamics of motion are defined exclusively with geometric constraints, in a process the authors refer to as "Kinematics from Shape".

Maarse (1987) performs a comparative study in which 14 different models of handwriting are considered. Handwriting is described as the combination of different movement primitives, which segment the handwriting trajectory in the spatial, velocity or acceleration domain. Such segments are delimited by "transition points" which are respectively located at minima and maxima of curvature, velocity minima, and zero crossings of acceleration. Bell-shaped velocity profiles are created with sinusoidal functions. The geometry of curved segments is described either by a circular arc, or by offsetting the motion in a direction perpendicular to the segment's principal direction. It is shown that the models describe writing units in the velocity domain, and that an *asymmetric* velocity profile gives the best reconstruction results.

3.5.3.2 Oscillatory models

Early models of handwriting employed analogue computing techniques to simulate handwritten traces. Van der Gon et al. (1962) developed an electronic device which simulated handwriting traces on an oscilloscope by regulating the timing of two second order components driven by rectangular acceleration pulses. Vredenbregt and Koster (1971) used a similar technique to build a writing machine driven by two DC motors. MacDonald (1966) fitted trapezoidal acceleration profiles to a recorded handwriting signal in order to replicate the handwritten traces on an oscilloscope screen. Dooijes (1983) generated handwriting with a model similar to the one developed by van der Gon et al. (1962), using two second order dynamical systems, excited by rectangular pulses. Hollerbach (1981) hypothesised that during handwriting movements muscles act as oscillating springs in the horizontal and vertical direction. This results in a model consisting of a pair of coupled horizontal and vertical oscillators that determine the velocity of a handwriting movement. While the model is capable of generating several simple instances of cursive script, with increasing complexity the number of parameters that have to

be adjusted becomes prohibitively large. Schomaker (1992) simulated simple handwriting movements with a biologically inspired neural network model of pulse oscillators. The neural network is trained on the recorded velocity of single letters of the alphabet, but while some of the results are satisfactory, at times the network does not converge. More recently, Nair and Hinton (2005) used a series of neural networks to learn the motion of two orthogonal mass spring systems from images of handwritten digits. The system is able to classify digits by extracting the corresponding "motor program" from a bitmap.

3.5.3.3 Ballistic models

On the basis of Plamondon's Kinematic Theory (Plamondon, 1995), the Sigma Lognormal (Plamondon et al., 2009) describes complex handwriting trajectories via the vectorial superposition of lognormal stroke primitives. With the assumption that curved handwriting movements are done by rotating the wrist, the direction and shape of strokes is described with a circular arc. The evolution of a stroke's curvature is given by using the time integral of the lognormal function (Eqn. 4.1) to interpolate between an initial and a final orientation. Complex trajectories can be described with the linear combination in time of a number of circular arc strokes. The Sigma Lognormal model has been extensively used in handwriting synthesis applications, including: handwritten signature synthesis (Plamondon et al., 2014; Ferrer et al., 2015), generating synthetic variations of a given handwriting specimens (Djioua and Plamondon, 2008a; Fischer et al., 2014) and CAPTCHA generation (Ramaiah et al., 2014). Plamondon and Privitera (1996) use a Self Organising Map (SOM) to learn a sparse sequence of ballistic targets, which is then used in conjunction with a Kinematic Theory model to generate trajectories.

Similarly to the Sigma Lognormal model, Bezine et al. (2004) develop a "Beta Elliptic" model, where handwriting trajectories are generated with the vectorial superposition of ballistic movement primitives using a beta function and an elliptic arc (rather than a Sigma Lognormal and circular arc). Ltaief et al. (2012) use a neural network to learn the mapping between the Beta Elliptic parameters and the resulting trajectory. The network is then used instead of the model as a trajectory generator. While the results are comparable to the use of the Sigma Lognormal model, there is less justification available (as a good model of human limb movements) and much less published material to compare with. In Chapter 9 we will use a mapping with the parameters of the Sigma Lognormal model to generate example-based trajectory stylisations.

3.5.3.4 Other methods

Del Vecchio et al. (2003) describe drawing motions via the combination of basic primitives that are referred to with the term *movemes*. A moveme is defined mathematically as a linear dynamical system that can uniquely describe a part of a motion. Stettiner and Chazan (1994) reproduce the planar velocity of handwritten characters with the impulse response of a slowly time varying second order linear system. Dynamical system parameters are fitted to uniformly sampled input examples using a Gaussian Mixture Model (GMM). The output is a distribution that can be stochastically sampled to generate variations of a handwritten character based on the input examples. In Chapter 5 we also present a method based on the combination of a linear system with a mixture of Gaussians. The method also outputs a trajectory distribution that can be stochastically sampled to generate variations according to the input data.

Bullock et al. (1993) develop the VITEWRITE (Vector Integration to Endoint WRITE) model, which

simulates neural signals to generate smooth handwriting trajectories that interpolate a motor plan made by a sequence of positions. Grossberg and Paine (2000) propose an extension to Bullock's model, AVITEWRITE (Adaptive VITEWRITE), which simulates visual attention and focus to adaptively learn handwriting trajectories by imitation. Both models attempt to accurately model the neural processes involved in handwriting production. The models successfully reproduce many effects seen in instances of real handwriting: such as power laws, isochrony, and also an increase in speed as learning progresses. Nevertheless, a lower performance in reproducing handwritten traces (Paine et al., 2004) and the architectural complexity of the models, limits their applicability for the task of style synthesis and computer graphics applications.

3.6 Letterform representation, generation and stylization

While modern fonts are commonly represented as a set of vector outlines (Karow, 1994), many of the visual conventions used in conceiving and creating a font or glyph can be traced back to their origins as *stroke*-based handwriting and calligraphy (Noordzij, 2005; Wang, 2013). The *stroke* is the fundamental primitve of calligraphy: the mark left by a gesture made with a pen, brush or another writing tool, from a starting position and moving towards a target. Tag-writing, at the origin of modern graffiti art, like calligraphy, is "handwriting pursued for its own sake, dedicated to the quality of shapes" (Noordzij, 2005), and it too is based on strokes. The outline of a glyph often conceals a latent structure as a set of generalized strokes that, when combined, closely reproduce the glyph's shape.

The wide adoption of outline-based representations can be traced back to the transition from hand produced letterforms to "punch-cutting" letterforms carved in steel, and the subsequent need to rapidly and efficiently convert existing type-faces to digital form. With this being the modern standard, the sustained contemporary interest in a stroke based representation can be traced back to the METAFONTsystem (Knuth, 1999). Donald Knuth defined a font description language that, in its first version, used the stroke metaphor to describe letterforms parametrically with a raster shape swept along a set of splines. Knuth (1999) demonstrated how this representation is useful to generate parametric variations of a letterform, ranging from different stroke weights, to more extreme effects that mimic handwriting or produce abstract letterforms. However, describing letterforms in MetaFont is not intuitive, and requires programming and mathematical knowledge that has limited its wider adoption in the visual design fields.

3.6.1 Structural representations of letterforms

Ghosh and Bigelow (1983) suggest that a MetaFont system should "provide a descriptive scheme in terms of which structural features of individual letter shapes can be *efficiently* described and talked about", where "efficiently" refers to the requirement that this description should be understandable both by a human (designer) and a computer. The authors suggest a representation of letterforms consisting of basic stroke-like primitives that can be assembled along a graph structure.

The variety of possible instantiations of a letterform includes variations of this structure, as well as variations of the way in which strokes are constructed along the structure. Hofstadter (1982) emphasises the possibly infinite variety of different structures and shapes that can result in a recognisable letterform. To study the process of letter creation in a tractable manner, he proposes a simplified letter

representation: a "grid-font" consisting of segments on a 3×7 grid. The project called "Letter Spirit" (Hofstadter et al., 1993; McGraw Jr, 1995; Rehling, 2001) is specifically aimed at modelling ways in which the human mind represents the concept of a letter and its style (or "spirit"). Letter analysis and genesis are implemented with a multi-agent system, where each agent (*"codelet"*) competes in the solution of a small cognitive sub-task. Each glyph is defined by its constituting parts called "roles" that are effectively similar to strokes. As an illustrative example, the letter "P" is described by two roles: one is a vertical bar, the other is a concave bowl-like profile with its extremities connected to the upper part of the vertical bar.

Cox et al. (1982) also propose an abstract graph-based description of letters simply called "skeleton", that distinguishes functional and structural components from the "embellishments" that produce different stylisation (in terms of strokes or parts, resulting in a given font). The skeleton graph edges can be either "explicit" (i.e. visible) or "implicit" (invisible), with the latter defining spatial constraints. Strokes are defined over a number of consecutive edges, and different stylisations are achieved by varying the stroke type. As an example, the letter "D" is made of two strokes, one vertical and one curved. The letter structure is defined with a series of relations:

- *Meet*: where two edges meet at a vertex. Can be executed with variable smoothness (e.g. a vertex located at the middle of the bowl of a "D", with a skeleton similar to a $\triangleright$)
- *Join*: where two edges meet at a vertex with a sharp corner (e.g. a "V")
- *Link*: where edges meet at a "T" junction.
- *Cross*: where two edges cross each other (e.g. such as for an "X")

In the fields of *semiotics*, Watt (1988) studies the evolution of the Latin alphabet with two complementary descriptions of the letterform: one *iconic* where the letter is described in its basic structure as a *sign*, and one *kinemic* — the study of gestures as body language — where the letter is considered as a dynamic representation of the movements that produce its trace on canvas. As an example, Watt demonstrates how the same iconic representation transforms an upper case "A" into a lower case "α" through a process he calls *facilitation*, which is the tendency to reduce effort during the kinemic production of a letter. While not well known in the computing domain, the combined dynamic and semantic representation of letters proposed by Watt offers interesting ideas for the computational synthesis of calligraphic art forms. The author also posits that the evolution of alphabets follows the evolutionary principles proposed by Lamarck (Burkhardt, 2013) rather then the more generally accepted Darwinian approach. A similar concept is separately proposed by Blanchard (1999) in the context of paleography with an "abstract ductus", equivalent to a schematic motor plan for a trace that can explain again the evolution of a capital "A" into "α".

In summary, a simple graph-based representation can be useful to distinguish different stylisations of a letter from its structure, where stylistic variations are given by different movements (Watt, 1988; Blanchard, 1999) or strokes and parts (Cox et al., 1982; Ghosh and Bigelow, 1983; Hu and Hersch, 2001) that are combined along this structure. In the next section we review different methods to generate strokes.

3.6.2 Stroke representations

The MetaFont system described different strokes with raster brush footprints swept along a set of parametric curves. Since then, many stroke generation methods have been developed, which can be used in conjunction with a structural representation of a letterform in order to render it and stylise it different ways. The computer graphics sub-field of Stroke Based Rendering (SBR) is aimed at mimicking the appearance of images made with painterly media. Many approaches have been proposed to simulate painterly brush strokes, for example raster based (Strassmann, 1986), texture synthesis (Yu and Peng, 2005), with a physical model of brush bristles (DiVerdi, 2013), fluid simulation (Curtis et al., 1997; Way et al., 2006), or with vector based methods (Hsu and Lee, 1994; Su et al., 2002). Scalera et al. (2017) mimic the appearance of spray paint with a Gaussian model of aerosol paint and ink deposition. With the aim of mimicking strokes in comics drawings, Saito et al. (2008) vary brush thickness as a function of curvature. These authors do not emphasise that their approach is consistent with the isogony principle, which in the kinematic domain relates to computing brush thickness as an inverse function of trajectory speed. Ferrer et al. (2015) exploit the smooth kinematics produced by the Delta Lognormal model to generate realistic ball-point pen strokes. Pen pressure and ink depositions are determined with an inverse function of trajectory speed. In Chapter 4 we also exploit the kinematics of the Sigma Lognormal model to develop a brush model that we use to render synthetic graffiti tags.

Yu and Peng (2005) employ a texture synthesis approach to render an abstract and expressive style of Chinese calligraphy (Cau Shu) which is characterised by rapid and expressive brush strokes. Different types of brush strokes are rendered by sweeping a deformable and rotating ellipse. A realistic rendition of brush patterns is achieved by re-synthesising collected calligraphic samples with Markov random fields (Cross and Jain, 1983). Lu et al. (2014) also use a texture synthesis approach to generate a decorative stylisation of a user defined path from pattern images.

The PostScript page description language is a widely adopted standard in various vector graphics applications, including electronic publishing and desktop publishing. It defines a *stroke* command that transforms a path into an outline. A PostScript stroke is defined by an envelope of constant width, different *caps* determining how a stroke ends and a *join* type determining how segments meet at a corner (rounded, bevel or miter). A *miter* joint connects sides of the envelope at their intersection. A *miter limit* defines a minimum angle threshold that avoids "spikes" when the angle at the joint is acute, and replaces the miter joint with a bevel.

Within a large body of work on control theoretic formulations of quintic B-splines, Fujioka, Kano et al. (Fujioka et al., 2006; Fujioka and Miyata, 2011) develop the so called "Dynamic Font Model", aimed at reproducing the strokes of Japanese calligraphy. B-spline control point locations are optimised so the resulting curve minimises a tradeoff between jerk magnitude and reconstruction error of an input trajectory. This allows variations in smoothness that can be used, for example, to mimic the appearance of cursive calligraphy or to generate smooth ligatures. Variably thick brush strokes are generated by using 3D trajectories and varying brush thickness based on the distance to a drawing surface. Seah et al. (2005) also develop an extension to B-splines that produces variably wide strokes with the addition of a coordinate that specifies a smoothly varying radius function. The method provides some desirable properties for CAGD applications, such as extensionality or the possibility to precisely compute intersections (Ao et al., 2018). In Chapter 5 we develop an optimal control method similar to the one of

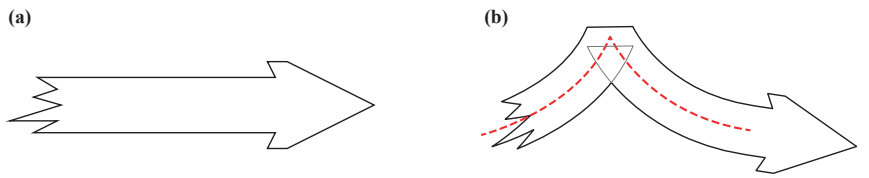


Figure 3.2: Example of a skeletal stroke. **(a)** An arrow-shaped prototype. **(b)** Deformation of the prototype along a spine (dashed red). The corner in the spine would produce a "fold" (gray), which is usually removed.

Fujioka and Miyata (2011), which is not based on B-splines but provides additional structure and flexibilities that are useful for our use case of graffiti synthesis. The method also supports smooth variably wide strokes similarly to Seah et al. (2005).

With the specific aim of font design, Jakubiak et al. (2006) describe a stroke model, consisting of path paired with a variably wide thickness profile and arbitrarily shaped caps. Schneider (2000) develops a similar method, but also defines a way to produce smooth blends where two strokes intersect. Hu and Hersch (2001) develop a component-based representation of fonts, consisting of parametric shape primitives like "bars", "serifs", "terminals" or "sweeps", where the latter are similar to strokes. The authors observe that when generating sweeps, it is desirable to offset the control points of a Bézier curve, rather than the curve itself, which produces visually more pleasing results.

Skeletal strokes (Hsu and Lee, 1994) is another widely adopted and flexible technique, which can reproduce strokes identical to the ones generated by PostScript but also a variety of other painterly and graphical effects. An input shape, called a *prototype*, is deformed along a destination path, called a *spine* (Figure 3.2.a). The deformation is performed by mapping portions of the prototype to portions of the spine, and then applying a deformation that depends on a variable width profile that maps distances along the spine to a pair of widths perpendicular to the spine (Figure 3.2.b).

3.6.2.1 Graffiti strokes: Self overlaps and layering.

One known issue with skeletal strokes is the appearance of folds in high curvature portions of the spine (Figure 3.2.b, in gray). This is generally considered undesirable, and several approaches have been proposed to adjust (Lang and Alexa, 2015) or to remove (Hsu and Lee, 1994; Asente, 2010) such features. Contrary to those, in Chapter 6, we exploit the folding behavior to mimic artistic self-overlapping effects that are often seen in graffiti art.

Differently from traditional type design and lettering, the strokes in graffiti pieces are often interlocked in complex ways and may have self-overlapping parts and loops (Ferri, 2016). Rather than combined with a simple union, they are superimposed and then traced, revealing an outline that is evocative of a 3D composition. Conventional stroking algorithms do not support these kinds of effects, which usually require a user to intervene by either masking parts of the outline (Wiley and Williams, 2006) or manually hiding the edges of a planar map representation of a drawing (Asente et al., 2007).

Wiley and Williams (2006) develop "Druid", a system for designing *interwoven* drawings. The system resolves overlaps between spline curves with a local labelling of crossings. However, our experi-

ments found their method unreliable in the presence of the folds and loops, such as the ones that are generated by the skeletal stroke algorithm (Figure 3.2.b). This can lead to edge visibility errors that propagate around the outline. A similar stability issue is known for certain hidden line removal approaches in 3D (Appel, 1967; Graphics and Applications, 1988). McCann and Pollard (2009) develop an interactive system for non-globally layering transparent bitmaps based on detecting regions of overlap, but it does not handle objects overlapping themselves. Igarashi and Mitani (2010) develop a similar method for 3D objects on a plane, which does permit self-overlaps. In Chapter 6 we develop a similar *local layering* method which operates on the outlines of 2D objects.

Asente et al. develop "LivePaint" (Asente et al., 2007), an interactive method for editing and painting planar maps (Baudelaire and Gangnet, 1989) that maintains the original underlying geometry. However, creating and modifying overlaps requires manually assigning appropriate stroke and fill attributes to edges and faces of the map. With a similar application in mind, Dalstein et al. developed Vector Graphics Complexes (VCG), a data structure specifically aimed at processing and editing potentially overlapping and intersecting vector art in a manner similar to planar maps, while maintaining topological and incidence relations (Dalstein et al., 2014). Our local layering implementation, in Chapter 6, also relies on planar maps, but in addition it maintains the structural information of a drawing across edits through a stroke-based representation.

3.7 Letterform stylisation and generation

In the previous sections we have implicitly argued that the combination of (i) a structural representation of a letter with (ii) a physiologically plausible model of trajectory formation paired with (iii) different stroke generation and rendering methods, can be used to generate a variety of stylisations of the letterform. In the first part of this thesis, we develop a set tools and primitives with the specific aim of implementing this framework for the case of graffiti and calligraphy. In the second part of the thesis, we seek to recover a structural representation and stroke primitives from existing geometry, in which information about a generative movement or an underlying stroke structure may be latent or unavailable. Recovering this underlying structure makes it easier to stylize and modify the input geometry in a way that would be difficult to achieve with image-based or geometric transformations alone. In addition, this procedure converts existing examples of tags, fonts, or other types of outlines, into a rich source of structures that can be used to procedurally generate graffiti and calligraphy in a variety of styles.

In the next sections we first review exiting works (summarised in Table 3.4) that have approached problems similar to ours, that of handwriting synthesis (Section 3.7.1) as well as calligraphy and font stylisation (Section 3.7.2). Some of the stylisation methods also rely on the recovery of strokes from character or glyph outlines, but most of these methods either require user assistance or assume a specific and restricted class of inputs. In order to automate this procedure, we consider concepts that have emerged in shape analysis and visual perception, and we review these in Section 3.8; these concepts help us build the foundation for the methods developed in part II of this thesis.

3.7.1 Handwriting synthesis

In handwriting analysis and synthesis, a distinction is usually made between *online* and *offline* data (Plamondon and Srihari, 2000). Online data consists of temporally ordered strokes,

References	Type	Structure	Strokes	Method	Vector
Graves (2013), Ha and Eck (2018), Tang et al. (2019)	handwriting/drawing	✓	✓	online	✓
Choi et al. (2003), Choi et al. (2004)	handwriting	✓	✓	online	✓
Haines et al. (2016)	handwriting	✗	✗	N.A.	✗
Chen et al. (2015)	handwriting	✗	✗	N.A.	✓
Wang et al. (2002), Wang et al. (2005)	handwriting	✗	✗	N.A.	✓
Lake et al. (2015)	handwriting	✓	✓	automatic	✓
Lee and Cho (1998)	handwriting	✓	✓	template-based	✓
Zhang and Liu (2009)	calligraphy	✗	✗	N.A	✓
Xu et al. (2009)	calligraphy	✓	✓	semi-automatic	✓
Lyu et al. (2017)	calligraphy	✗	✗	N.A.	✗
Miyazaki et al. (2019)	fonts/calligraphy	✓	✓	template-based	✓
Rehling (2001), Grebert et al. (1992)	fonts	✓	✗	N.A.	✓
Tenenbaum and Freeman (2000)	fonts	✗	✗	N.A.	✗
Lian and Xiao (2012)	fonts	✓	✓	template-based	✓
Suveeranont and Igarashi (2010)	fonts	✓	✓	template-based	✓
Campbell and Kautz (2014)	fonts	✗	✗	N.A.	✓
Lopes et al. (2019)	fonts	✗	✗	N.A.	✓
Phan et al. (2015)	fonts	✓	✓	user-guided	✓
Wang et al. (2020)	fonts	✗	✗	N.A.	✗
Zhang et al. (2017a)	fonts	✓	✓	user-guided	✓
Xu and Kaplan (2007)	fonts	✓	✗	user-guided	✓
Zou et al. (2016)	fonts	✓	✗	automatic	✓

Table 3.4: Font/calligraphy/handwriting generation and synthesis.

where each stroke is usually encoded as a sequence of 2D points. Offline data consists of images or curves, where the stroke temporal order and structure is unavailable.

In one popular handwriting synthesis approach, a system generates handwritten strings from a few sentences written by a user, which are digitised in an online (Wang et al., 2005; Lian et al., 2018) or offline (Chen et al., 2015; Haines et al., 2016) form. The sentences are required to cover a sufficient variety of characters of the alphabet. These approaches assume a mapping between the input examples and the corresponding text, which for sufficiently readable handwriting, can be done with line and character segmentation (Haines et al., 2016) or with a handwriting recogniser (Wang et al., 2005). This can be challenging for our use cases of graffiti tags or calligraphy, which are usually highly stylised signatures that are often unreadable (as meaningful text) by an untrained human.

A different approach consists in generating handwriting by concatenating a series of predefined motor plans, one per letter of the alphabet. Different stylised trajectories are then generated with the simulation of a movement that follows the concatenated motor plans. Lee and Cho (1998) use this approach to synthesise Korean handwriting with a Beta Elliptic

model, while Ferrer et al. (2015) use a similar approach to generate synthetic signatures with a combination of filtering and the Delta Lognormal model. The methods we will demonstrate in the following chapters can all be applied to concatenated templates in a similar manner.

The Sigma Lognormal model of handwriting has been widely used to generate realistic variations of handwriting, with applications that vary from the generation of synthetic signatures (Galbally et al., 2012) to the generation of handwritten CAPTCHAs (Ramaiah et al., 2014). These methods rely on a reconstruction procedure that recovers the needed parameters from the velocity and geometry of a digitised trace. This can be done with a number of state of the art methods (O'Reilly and Plamondon, 2008; Plamondon et al., 2014; Fischer et al., 2014; Ferrer et al., 2018), but all these assume that the input already encodes a velocity profile, thus limiting their applicability to digitised movements only. In Chapter 8 we develop a novel method that infers Sigma Lognormal parameters from the geometry of an arbitrary trace, which effectively recovers a latent generative movement. We then use this reconstruction in combination with a *Recurrent Neural Network* (RNN) to generate example-driven stylisations of handwriting, tags or other kinds of inputs in vector form.

Recently, a growing number of methods have used RNNs to synthesise handwriting (Graves, 2013; Zhang et al., 2017b; Tang et al., 2019) as well as drawings (Ha and Eck, 2018). These models are usually trained on rather large training sets consisting of online handwriting data in the form of digitised point sequences. As a result, the predictions of the networks also consist of temporally ordered points, which mimic the movements followed by a writing pen. It is interesting to note that the more recent methods (Ha and Eck, 2018; Zhang et al., 2017b; Tang et al., 2019) all use polyline simplification to simplify the dense training trajectories into sparse sequences of points, suggesting that this improves training speed and robustness. However, the authors do not note that this simplification is similar in practice to a motor plan the describes the fine evolution of a trajectory. We investigate this concept more in depth in Chapter 9, where we explore the combination of an RNN model similar to the one developed by Graves (2013), with a representation of movement in terms of the Sigma Lognormal model.

3.7.2 Font and calligraphy generation and stylisation

One well studied problem in the domain of font generation and stylisation, was initially proposed by Hofstadter et al. (1993) in the "Letter Spirit" project: given a few exemplars of the letters of a font, generate all the alphabet with a style (or "spirit") that is consistent with the exemplars. The letter spirit project culminated with the thesis of Rehling (2001), who described a complex cognitively inspired architecture that generated complete grid fonts with a style that is similar to a few examples given by a user. In the meantime, Grebert et al. (1992) approached the same problem with a simpler architecture consisting of a three layer neural network. The network perfectly reproduced the letters it was trained on and generated the

missing letters with similar regularities to the input exemplars, but not a necessarily readable structure (Rehling, 2001).

The same kind of problem has been approached also with bitmap representations of fonts. Tenenbaum and Freeman (2000) developed a similar system using a bilinear model and a vector field representation of glyph rasters. More recently, a number of Deep Learning methods have been proposed which can predict the missing glyphs for instances of fonts (Azadi et al., 2018; Hayashi et al., 2019; Gao et al., 2019) as well as calligraphy (Lyu et al., 2017; Wen et al., 2019). One common limitation of these methods is that they produce bitmap outputs with a relatively low resolution. In practice, these bitmaps can be vectorised, but this would still limit the possibilities of editing and varying the results in a meaningful manner.

Hofstadter et al. (1993) use fonts as an example of the vast range of structures and shapes that a recognisable letterform can assume, and to demonstrate the challenge of modeling the creative process underlying letter stylisation and design. At the same time, if we temporarily ignore the complexity of this creative process, fonts can also be seen as an equally varied *source* of letter structures and shapes in a variety of styles, languages and writing systems. Indeed a number of methods use font outlines as a starting point to generate stylised letterforms.

Some methods approach font and calligraphy stylisation as an outline correspondence problem. Zhang and Liu (2009) synthesise new calligraphy samples by employing an energy minimising deformation between the outlines of different character samples. Campbell and Kautz (2014) generate a latent manifold from font outlines with the same topological structure and generate new fonts by interpolating and extrapolating points on the manifold. Balashova et al. (2019) use a template driven approach to decompose an outline into parts that they also call strokes. This provides additional structure that allows to apply a method similar to the one of Campbell and Kautz (2014) also to outlines with different topological structures. Such outline-based methods do not provide sufficient control to mimic the variety of stylisations that can be observed in graffiti art.

Other methods, adopt a similar correspondence-based approach but operate at the level of strokes. Depending on the method, strokes are either extracted automatically (Xu et al., 2009; Lian et al., 2018) from a font outline or with a user guided (Phan et al., 2015) or template-based procedure (Suveeranont and Igarashi, 2010). Xu et al. (2009) use a semi-automatic procedure to decompose example outlines into strokes, and then generate calligraphic stylisations of a character with a weighted interpolation or extrapolation between strokes. The strokes are then rendered with a realistic brush model. Xu et al. (2012) use feedback from 5 expert calligraphers to train a neural network that automatically evaluates the results of the system (Xu et al., 2009) and that is used to automatically tune the stylisation parameters. Lian et al. (2018) also use an automatic segmentation into strokes to generate Chinese handwriting and calligraphy stylisations from examples. Suveeranont and Igarashi (2010) develop a

method similar to the one proposed by Xu et al. (2009) but extended to arbitrary font outlines. However the stroke segmentation relies on a set of predefined templates. Phan et al. (2015) rely on a user guided stroke decomposition to solve a problem similar to the one proposed by (Hofstadter et al., 1993) and use a manifold similar to the one proposed by Campbell and Kautz (2014) but computed for segmented strokes.

All these methods approach font generation and stylisation exclusively as a style transfer from one ore more example glyphs to an entire font. Different approaches exist, for example Zhang et al. (2017a) use simple user defined sketches to segment words written in a given font into strokes and then reconstruct the word with shapes that are semantically similar to the written word. Xu and Kaplan (2007) and Zou et al. (2016) automatically generate "calligrams" consisting of letters that are deformed to fit inside a given shape outline. Zou et al. (2016) guide the packing procedure by decomposing the outline into parts (Luo et al., 2015) by manually identifying protrusions in the letter that are effectively similar to strokes.

3.7.3 Stroke segmentation

The problem of decomposing an outline into strokes is challenging since it involves disambiguating regions where multiple strokes may cross or overlap. This kind of problem has been addressed by many methods for the traces of handwriting (Plamondon and Privitera, 1999; Lake et al., 2015) or drawing (Favreau et al., 2016). However, these methods assume an input produced with uniformly thin strokes, which is often not the case for fonts.

Of the methods discussed in the previous section, the only ones that rely on automatic segmentation are focused on Chinese characters. The problem of stroke segmentation for Chinese, or more generally east Asian characters, is well developed. Decomposing East Asian characters, which are often based on a hierarchical structure of radicals and strokes, has been well studied (Wang et al., 2002; Sun et al., 2014; Chen et al., 2017) and extensive datasets are available for data-driven methods. For example, Kim et al. (2018) train a neural network on the "makemeahanzi" (Kishore, 2018) dataset, resulting in a data-driven method that decomposes similar characters into potentially overlapping strokes. The segmentation method of Lian et al. (2018) is automatic and robust, but it also exploits the precise hierarchical structure of Chinese characters and relies on a dataset of a significant number (27,533) of manually labelled and categorised characters.

These same methods usually fail with Western fonts and glyphs, which have a wider range of stylistic variations and decorations, and which often blend components into each other in ways that make segmentation difficult. This ill-posed problem (Lamiroy et al., 2015) has been partially addressed with user-defined templates (Herz et al., 1997; Suveeranont and Igarashi, 2010; Phan et al., 2015; Zhang et al., 2017a; Balashova et al., 2019) or a detailed analysis of glyph outlines (Shamir and Rappoport, 1996). In particular, the method of Shamir and Rappoport (1996) automatically identifies typographic features such as serifs, bars and

stems. However, it does not disambiguate more complex cases where multiple strokes intersect. The method relies on the identification of salient features along an outline, such as extrema (of curvature), inflections, cusps, corners and relatively straight regions. In Chapter 11 we introduce a related geometric approach, while in the next section we review the foundations that inform its implementation.

3.8 From shape to strokes

The term "shape" is used in everyday language, with expressions such as "the shape of a tree", "a spherical shape", "an abstract polygonal shape", which seldom require any additional definitions to be intuitively understood. At the same time, a precise definition of shape is ill posed (Koenderink, 1990), and it is rarely found in the computational literature, where the assumption is usually made that the term refers to some geometric description of an object or its outline, let it be contour samples or pixels in 2D, or triangles or voxels in 3D. Computational geometry, computer graphics and mathematics are full of useful, and different "shape representations", and a precise definition of the term is useful for their categorization and to understand how they are related. Arnheim (1954) defines shape as an "active occupation" of the mind (p. 43):

> "..in looking at an object, we reach out for it. With an invisible finger we move through the space around us, go out to the distant places where things are found, touch them, catch them, scan their surfaces, trace their borders, explore their texture."

A related, but more precise definition is proposed by Koenderink (1990), who defines shape "operationally" as the structure resulting from a series of measurements made in a "field" surrounding an object's outline. Shape depends as much on the object as it depends on the method that is used to "probe" this field. Leymarie (2006) takes this definition further, by viewing shape representations as sequences of transforms, which emphasise or reveal certain properties of an object, while de-emphasising or discarding others. Transform sequences can be organized into two main categories: *horizontal transform sequences* that act *along* an outline, and *vertical transform sequences* that act *perpendicular to* an outline. This dichotomy bears similarities to the one, commonly used for two dimensional shape, of *contour based geometry* and *region based geometry* (Singh, 2015). A sufficiently informative shape representation can be used to characterise an object's *morphology*, that is to study its form intended as perceived visual qualities such as "roundedness", "sharpness" (Leymarie, 2006; Albertazzi, 2019).

A countless number of shape representation methods have been developed in the domains of computer vision and graphics. A great majority of these methods is aimed at providing quantitative measures in terms of dimensionless quantities that are useful for recognition

or shape retrieval tasks (Zhang and Lu, 2004; Rosin, 2005), but are less useful for our goal of shape analysis for generative tasks such as stylisation and abstraction.

It is a commonly held view that the process of drawing or painting is like "learning to see" (Koenderink and van Doorn, 2008), and it can be argued that a system aimed at mimicking a visual art form should rely on representations and principles that are consistent with the ones that are at the basis of the human visual system (Mi, 2006). As a result, the following sections will narrow the focus on shape representations that have some form of perceptual grounding, building up from curvature based shape representations (Section 3.8.1) to symmetry based shape representations (Section 3.8.2), to principles of perceptual grouping and contour integration (Section 3.8.3). These representations are the basis for a higher level descriptions of shape, in terms of perceptually meaningful parts (Section 3.8.4) and ultimately in term of strokes, which we will recover in Chapter 10.

3.8.1 Curvature based shape representations

The study of the curvature along a contour has been the focus of decades of research in various fields, including visual perception and cognitive science, as well as computer vision and pattern recognition. *Curvature extrema* are perceptually important points (De Winter and Wagemans, 2008b) at which the curvature function reaches a local minimum or maximum. For regular 2D curves, there are four types of curvature extrema, which are often denoted as $M+, m-, M-, m+$ (Richards and Hoffman, 1985; Leyton, 1987; De Winter and Wagemans, 2008b), where $M+$ and $m-$ respectively denote *positive maxima* and *negative minima* of curvature, while $M-$ and $m+$ respectively denote *negative maxima* and *positive minima* of curvature (Figure 3.3). The first pair of extrema types ($M+, m-$) are *absolute maxima* of the curvature function that correspond to "sharper" features (turns, bends) and are the ones typically taken into consideration in methods aimed at identifying curvature extrema. The second pair of extrema types ($M-, m+$) map to *absolute minima* of the curvature function and thus correspond to flatter or squashed or compressed shape regions (Leyton, 1988). For piecewise continuous curves, corners and cusps can be viewed as the limits to positive and negative infinity of curvature for absolute maxima ($M+, m-$) (Leyton, 2006). *Inflections* are the locations where the curvature function changes sign (e.g. the midpoint of an "S"-like curve) and are also considered to be perceptually significant (De Winter and Wagemans, 2008b) or generally useful for the description of shape (Richards and Hoffman, 1985).

3.8.1.1 Perception of curvature

As early as 1954, before the advent of computers as a mainstream tool, Attneave (1954) experimentally argued that shape information is concentrated at object contours and especially at corners and absolute maxima of curvature along these contours. This led him to draw a now famous picture, known as "Attneave's cat", which depicts a clearly recognizable cat by connecting absolute curvature maxima along its silhouette with straight line segments. More

recently, Feldman and Singh (2005) derived a more precise information-theoretical formulation of Attneave's claims for discrete traces by defining information content as the negative log-likelihood, or *surprisal*, of deviations from straightness, measured with a von Mises distribution (a circular analogue of a Gaussian) over the turning angles ϕ. For simple and closed contours, Feldman and Singh (2005) suggest that turns towards the interior should be considered more probable, which gives a higher information content to negative minima ($m-$), a result that is consistent with the hypothesis of a perceptual bias towards concavity (Hoffman and Singh, 1997; Hulleman et al., 2000). The perceptual importance of curvature extrema is confirmed experimentally by De Winter and Wagemans (2008b) by involving a large (for this type of study) number ($N = 161$) of participants, that are asked to label perceptually salient points along a sufficiently large set ($N = 260$) of object boundaries. The experimental results, however, do not show the preference for concavities suggested by Feldman and Singh (2005) and others, a result that is attributed to the influence of non-local factors on the preferences shown by participants. Interestingly, the magnitude of curvature alone is not found to be a good predictor of these preferences. In their study, (De Winter and Wagemans, 2008b) also evaluate a number of measures of *saliency* for curvature extrema, including a variant of the turning angle measure proposed by Feldman and Singh (2005). Table 3.5 gives an overview of different measures.

Measure	**Short Description**	**References**	**Cor.**
Inverse compactness	Part contour length squared divided by part area	De Winter and Wagemans (2008b), Zusne (1970)	≈ 0.6
Relative size	Part area divided by total area	De Winter and Wagemans (2008b), Hoffman and Singh (1997)	≈ 0.45
Stick-out	Length of part divided by length of part base	De Winter and Wagemans (2008b), Hoffman and Singh (1997)	≈ 0.4
Absolute curvature	Absolute magnitude of curvature	De Winter and Wagemans (2008b)	≈ 0.3
Turning angle	Angle between perpendiculars to flanking segments	De Winter and Wagemans (2008b)	≈ 0.85
Surprisal	Negative log-likelihood of turning angle probability	Feldman and Singh (2005)	NA
Circular arc length	$(\phi l_1 l_2) / (l_1 + l_2)$	Latecki and Lakämper (1998)	NA

Table 3.5: Curvature saliency measures. Here a "part" is a contour segment containing the extremum and delimited by the two adjacent extrema, the "flanking segments" are the segments with length l_1, l_2 going from the extremum to the adjacent extrema, and ϕ is the absolute angle between the normals to the flanking segments. The correlation values are the (approximate) maximum values from the evaluation in the results of De Winter and Wagemans (2008b) (refer to Figure 7 of the paper).

Visual search studies[8] suggest that curvature representations occur pre-attentively (Treisman and Gormican, 1988; Hulleman et al., 2000), that is early in the human visual system, before attention is focused on a certain region of the visual field (Wagemans et al., 2012). At a higher level, curvature plays an important role in the perceptual decomposition of objects into parts (Richards and Hoffman, 1985; Brault and Plamondon, 1993a; De Winter and Wagemans, 2006) and the characterization of these parts (e.g. bends, necks), a subject that will be described more precisely in Section 3.8.4. As we have previously seen, curvature also plays an important role in the characterisation of human movement and handwriting by relating the kinematics of a movement to its geometry with inverse relations such as the power law (Viviani and Schneider, 1991). These relations suggest that curvature is a potential perceptual cue for the mental recovery of a generating movement from the geometry of its trace (Pignocchi, 2010).

3.8.1.2 Digital curvature

A robust identification of curvature extrema and corners, or *dominant points*, for digitally sampled contours can be difficult as it requires the evaluation of a second order differential quantity, which tends to amplify the effects of digitization noise in the input. One popular method to overcome this problem is to first smooth the digitised signal using a filter (e.g convolving with a Gaussian) or with some analytic function (e.g. smoothing splines) followed by a peak finding method (Leymarie and Levine, 1989). However, this risks removing perceptually important features and choosing parameters that function well across a large range of inputs remains a difficult task. One approach to address this issue is to construct a *scale-space* (Witkin, 1983; Koenderink, 1984), in which features that are tracked across multiple scales are considered more significant. Such a scale-space is very often produced by iterative Gaussian filtering in the spatial domain, or via the frequency domain using wavelets (De Stefano et al., 2005). While this approach is robust to noise, because of smoothing, it typically fails to provide an accurate estimate of curvature or to detect localised features such as segments of approximately constant curvature (e.g circular arc segments), which are hypothesised to play an important role in contour perception (Garrigan and Kellman, 2011). To address feature localisation, Leymarie and Levine (1989) propose a *structural* notion of scale using morphological operations (opening, closing) over the curvature function, resulting in its approximation in terms of line segments.

In practice, the concept of scale defined as an implicit characteristic of a curvature function along a trace was really an attempt to attach a notion of a *region of support* (Teh and Chin, 1989) to a feature, such that significant features can be characterised in part by how much of a segment of the trace being traversed they can represent. A number of methods identify such

[8] A visual search experimental paradigm is often used to evaluate if a given feature is computed pre-attentively. A rapid reaction time of a search for the feature among a large number of distractors and a slow reaction time for the reverse suggest that the feature is processed pre-attentively.

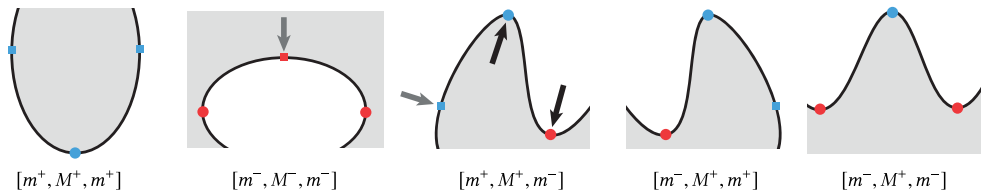


Figure 3.3: Codon types according to Richards and Hoffman (1985), together with the corresponding triplets of curvature extrema: $\bullet m-$, $\blacksquare m+$, $\blacksquare M-$ and $\bullet M+$. The gray and black arrows are not part of a codon's definition, but use the curvature extrema to demonstrate "process arrows", as proposed by Leyton (1988) in his process grammar. The black arrows point at absolute maxima of curvature ($m-, M+$) and indicate a process that produces an indentation (from the exterior) or a protrusion (from the interior) along the outline. The gray arrows point at absolute minima of curvature ($m-, M+$) and indicate a process that "squashes" (from the exterior) or exherts resistance (from the interior) along the outline.

support regions with an iterative traversal of trace segments surrounding a given point (eg. Brault and Plamondon, 1993a), effectively resulting in a hybrid between a horizontal and a vertical process. Sarfraz (2008, Chapters 11 and 12) reviews and provides implementation details for a number of these methods, including a method developed by the author himself (Sarfraz et al., 2006). A particularly useful example of such methods is the so-called Discrete Curve Evolution (DCE) (Latecki and Lakämper, 1998), where a polygonal reconstruction of a trace is incrementally refined by adding *dominant points* determined based on a salience measure computed as the length of two flanking support segments and the turning angle between these.

3.8.1.3 Curvature based shape descriptors

A number of methods organize contour-segments based on an analysis of curvature. With the aim of object recognition, Asada and Brady (1986) describe a "curvature primal sketch" that organizes curvature extrema into a grammar describing shape morphology, where single extrema categorised as "corners" and "smooth bends" and groups of extrema are further categorised as "cranks", "ends" and "bumps". Richards and Hoffman (1985) propose a partitioning of closed contours into five features types called "codons" (Figure 3.3), contour segments defined by curvature extrema triplets characterised by a central curvature maximum ($M+$ or $M-$) bounded by two curvature minima ($m-$ or $m+$). Rosin (1993) notes that codons do not allow for curvature discontinuities, and thus he extends the representation to include cusps and corners. The same issue with codons is noted by Galton and Meathrel (1999), who propose a partitioning of contours based on a grammar of "curvature types" (e.g. line segments, concave/convex curve segments, cusps). Kellman and Garrigan (2007) propose that contour segments with constant curvature (i.e. circular arc segments, or "arclets") are a feature detected early in the vision process which improves recognition performance (Garrigan

and Kellman, 2011).

Leyton (1988) proposes a conceptually different interpretation of curvature extrema with his "process grammar", where curvature extrema are viewed as the result of a deformation process producing a protrusion/indentation ($M+, m-$) or exerting a squashing/resistive force ($m+, M-$) on a shape outline (Figure 3.3). The process grammar describes two types of transformation rules, "continuation" and "bifurcation", which describe how extrema are created or vary during a deformation. For a concise overview of these rules refer to Leymarie (2006).

3.8.2 Axial symmetry based shape representations

An important alternative to focusing solely on curvature as an intermediate representation, also proposed early on by the computing community, is instead to rely on a generalization of the symmetry axis for shape. Symmetry is the invariance of an object under a class of transformations, and, in two dimensions, a symmetry axis conventionally denotes a straight line that delimits two sides of a shape that are congruent under a reflection along the line. This concept can be generalized to capture point-wise symmetry relations, resulting in a graph of potentially curved axes that is commonly known as the Symmetry Axis Transform (SAT).[9] Originally pioneered by Harry Blum for the description of biological shape (Blum, 1962), the SAT is a shape representation that provides a bridge between a shape's geometry and topology (Blum, 1973). It consists of a *symmetry axis*, the centers of "maximally inscribed disks", paired with a *radius function* that maps the centers to the corresponding disk radii. The symmetry axis consists of one or more potentially curved axial segments, often referred to as *branches* (Macrini et al., 2011; Shaked and Bruckstein, 1998), which can be organised as a directed graph. For closed contours, the SAT is a *complete shape representation* since a solid object can be fully reconstructed with a union of the maximal disks contained within its interior (Blum, 1973). Wolter (1992) and collaborators (Sherbrooke et al., 1996; Wolter and Friese, 2000) show that the SAT of the interior is an *homotopy equivalent* of the object as a solid,[10] meaning that it describes an equivalent topology, with the advantage of doing so with a more compact representation (Tagliasacchi, 2013). Experimental evidence suggests that SAT like representations are likely to be part of the "machinery" used by the human brain to perceive (Kovács et al., 1998; Kimia, 2003; Firestone and Scholl, 2014) and to recognize (Ayzenberg and Lourenco, 2019) shapes.

3.8.2.1 SAT: definitions

Many equivalent definitions of the SAT exist emphasising different properties and leading to different means of implementing the transform (Tagliasacchi, 2013):

[9] Other popular names found in the literature include: Medial Axis, Skeleton, Shock graph (in 2D) and Shock scaffold (in 3D).

[10] This homotopy relationship was later demonstrated again by Lieutier (2003).

- *Maximally inscribed disks*: A commonly used definition is restricted to the interior of closed contours, i.e. for solids, and considers the SAT as the union of centers of *maximally inscribed* disks(or balls in 3D) together with their associated radii. This definition is often the basis for a raster based computation of the SAT, which can be done for example with morphological thinning or through a distance transform (Leymarie and Levine, 1992). The popularity of this definition in the literature has led to the common misconception that the SAT is only defined for solids, while already in its early formulation (Blum, 1967, 1973) the transform is defined for open segments as well as isolated points or samples (making it directly related to Voronoi diagrams, since commonly used in CGAD).

- *Grassfire analogy*: A definition of the SAT that generalizes well to the case of open traces is based on the "prairie grassfire" or meeting or collapsing wavefront analogy, in which the maximal disk centers are given by the "quench" points (Leymarie and Levine, 1992), or *shocks* (Kimia et al., 1995), at which fire fronts or waves propagating from the object boundary meet and stop expanding. This definition can be implemented by following the ridges of the height surface produced by a reaction process evolving over a discrete lattice and initiated at points sampled along a trace or object boundary (Leymarie and Levine, 1992; Kimia et al., 1995; Gao et al., 2018).

- *Maximal bitangent disks*: With yet another (closely related) definition, the SAT is the locus of maximal *bitangent* disks, which assumes an input consisting of smooth outline segments built from (at least) twice differentiable curves. This provides the basis for an analysis of the SAT under the lens of differential geometry . This definition can be extended to accommodate for the end-points of open traces or breaks in curvature (cusps, corners), for example, by defining a set of *radials* that interpolate along gaps where normals do not exist (Blum, 1973) and results in an extended set of "pannormals" that includes both the sets of normals and radials. This allows to relax the bi-tangency constraint to one of *radial contact* and leads to a SAT definition in which each disk center is *equidistant* to at least two distinct trace points that are closer to the disk center than to any other trace point. This definition permits a SAT implementation in terms of a *Voronoi* diagram, which, after Blum introduced his ideas in the 1960's and 1970's, became widely used in the computational geometry (and CGAD) literature.

- *Vornoi based methods*: The (2D) Voronoi diagram of a set of points (or sites) consists of a planar graph that partitions the plane into convex regions (a.k.a. Voronoi regions) that are nearest to each site. Each (Voronoi) edge of the graph bisects two sites that have generated it, and all the points along the edge are equidistant from the two sites. It is thus possible to construct a disk that is centred along the edge, is tangent to the

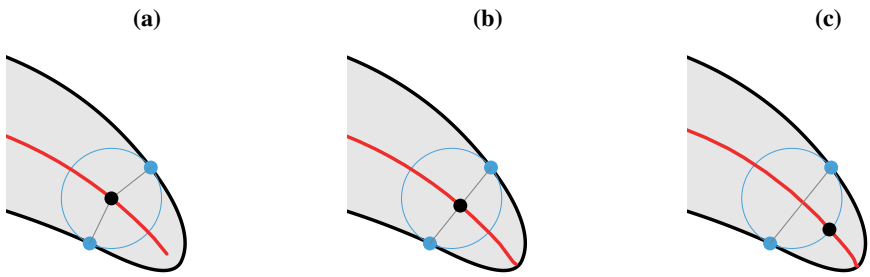


Figure 3.4: Some symmetry axis variants (red), showing one of the maximal disks (blue circle) , with two symmetric points along the outline (blue dots) and the axis point generated by the disk (black dot). **(a)** Blum's SAT is the locus of disk centers. **(b)** Brady and Asada's SLS is the locus of chord midpoints. **(c)** Leyton's PISA is the locus of the shortest arc midpoint.

two generating sites and does not contain any other site. The vertices of the Voronoi diagram are equidistant from 3 sites (in general positions) and correspond with the circumcenter of a triangle connecting the sites that does not contain any other sites. The set of all such triangles defines the *dual* of the Voronoi diagram, which is known as the *Delaunay* triangulation of the sites. If we consider the disks centred at Voronoi edges and vertices, this becomes equivalent to the maximal SAT disks introduced by Blum in the 1960's.[11] Similarly to the SAT, one common analogy for computing the Voronoi diagram is the one of fire fronts uniformly propagating from each site: the points at which the fire fronts first meet (and extinguish) are the edges of the Voronoi diagram (O'Rourke, 1998). The (2D) Voronoi diagram is very well studied in the computational geometry domain, with many robust implementations that usually compute the diagram $O(n \log n)$ time (O'Rourke, 1998). As a result it has become one of the most popular and practical methods to approximate the SAT for discrete (sampled) traces inputs (Ogniewicz, 1992; Amenta and Bern, 1999; Durix et al., 2019).

3.8.2.2 SAT variants and extensions

Many extensions and variants of the SAT have been developed since its inception (Figure 3.4), a number of which were already suggested by Harry Blum in his seminal long articles (Blum, 1967, 1973). Kimia et al. (1995) use a reaction-diffusion process initiated along a contour to construct symmetry axes in terms of singularities or shocks, which occur at the disk centers of Blum's SAT. Shocks carry additional information which depends on the variation of the

[11] Blum's ideas and definitions in fact correspond to what is now called the Generalised Voronoi diagram for sites that can be points, open curve segments, solids, or combination thereof. Blum and his collaborators pioneered these concepts through the 1960's and 1970's both for 2D and 3D sets of sites.

radius function, and this is used to construct a grammar of four shock types that are useful for object understanding and recognition (e.g. indicating neck loci where one may split an object in parts).[12]

In a wavefront propagation setting, the SAT is given by the points at which two concentric wavefronts initiated along a trace first meet. Blum (1967) calls this a "blocked" symmetry set and observes that additional symmetry axes can be identified by letting the propagation continue. This results in an "unblocked" symmetry set, which is also known as *full symmetry set* (FSS) and has been extensively studied mathematically, in particlar by mathematician Peter Giblin and collaborators (Giblin, 2000). The FSS captures additional symmetries of a shape that cannot be computed with the SAT. As a simple example, consider the case of a vertically oriented ellipse: the SAT produces a single vertical symmetry axis, while the FSS produces an horizontal and a vertical one. While these additional symmetries may be practically useful, in practice the FSS is challenging to compute and, with increasing shape complexity, it produces a large amount of axes that can become impractical to manage and interpret or use in applications.

A similar complexity issue arises with the *smooth local symmetries* (SLS) (Brady and Asada, 1984), a variant of the SAT already noted earlier by Blum (1973), in which symmetry axis points are located at disk chord midpoints rather than centers. The SLS also generates two axes for the case of the ellipse, with the axes completely contained within its interior, and for certain classes of shapes it can produce symmetry axes that are closer to the perceived symmetric structure of an object when compared with the SAT (Brady and Asada, 1984, Fig. 8). However, the SLS does not maintain shape topology, and similarly to the FSS, it can become impractically complex for more complicated shapes. Mi and DeCarlo (2007) overcome these complexity issues by only computing subsets of the SLS starting from a digital estimate of curvature extrema, and then using the resulting axes to decompose objects into potential parts.

With the aim of defining a process-based grammar of curvature extrema, Leyton (1988) derives the *process inferring symmetry axis* (PISA), a variant of the SLS in which symmetry axis points are located at maximal disk arc midpoints. While the PISA is conceptually useful in the framework of Leyton's theories of shape (Leyton, 2001b, 2006), its definition remains theoretical. The methods summarized here are the ones most relevant to the work presented in this thesis, but this list is not exhaustive. Many other symmetry axis variants exist, for example extensions to three dimensions (Leymarie and Kimia, 2001, 2007; Bucksch and Lindenbergh, 2008). For a more extensive review the interested reader is referred to the recent surveys by Tagliasacchi (2013) and Saha et al. (2016), as well as the book on medial representations edited by Siddiqi and Pizer (2008), and for evidence of SAT-like representations being studied

[12] Montanari (1969) defines a similar concept as shocks and their hierarchy, called *breakpoints* (initial, intermediate, final) in relation to the speed of propagation of SAT formation along its edges. Blum (1973) refers to this concept.

across the arts, physiology, perception and computing, refer to the recent article by Leymarie and Aparajeya (2017).

3.8.2.3 Relationship to curvature

It has been known since early developments (Blum and Nagel, 1978), that the symmetry axis endpoints can coincide with maxima of absolute curvature or corners of a trace, a property that also holds for SAT variants such as the FSS, SLS and PISA (Leyton, 1987). Indeed, the identification of curvature extrema along a contour has been widely used as a starting point for computing the SAT in its original formulation (Leymarie and Levine, 1992), as well as the SLS (Mi and DeCarlo, 2007). Leyton (1987) formalizes the relation between curvature extrema and symmetry axes with the "symmetry curvature duality" theorem, which states that any (smooth) trace segment bounded by two curvature extrema of the same type has a unique symmetry axis that terminates at an extrema of the opposite type. Here "type" stands for the extremum being a *signed* minimum ($m-, M-$) or maximum ($m+, M+$) of curvature (Figure 3.3). Leyton relies on the SLS to prove his result, but shows that it holds also for the SAT and PISA, with the exception of absolute minima ($m+, M-$), for which the SAT produces no symmetry axes. For the case of absolute minima, Leyton proposes a theoretical variant of the SAT that he denotes as Exscribed Symmetry Axis Transform (ESAT). The ESAT can be considered a "dual" of the maximally inscribed disk SAT definition, for which we replace "maximally" with "minimally" and "inscribed" with "exscribed/circumscribed", that is the loci of all minimally cirumscribed disks to the contour.

Hayes and Leyton (1989) and later Leyton (2006), extend the validity of these results to the case of breaks in curvature (corners and cusps). With a related result, Kimmel et al. (1995) show that any contour segment bounded by two generating SAT points always contains at least one absolute curvature maximum. Symmetry-curvature duality provides the basis for Leyton's process grammar, and leads to the definition of the "interaction principle" (Leyton, 1989), stating that symmetry axes terminating at the extrema can be interpreted as the directions along which these processes have acted, and thus provide a mean to recover a (plausible) "history" of the processes that give rise to a shape, starting from a circle that is seen as perfectly symmetric "primordial egg" (Koenderink, 1990). This has led to the development of a group theoretic "generative theory of shape" (Leyton, 2001b), which Leyton has applied to the analysis of paintings (Leyton, 2006) as well as architecture (Leyton, 2001a). While these results are conceptually very interesting with respect to the analysis and generation of graffiti art, Hendrickx and Wagemans (1999) have questioned the mathematical and perceptual soundness of Leyton's group theoretical work. However, the authors confirm the correctness of early results such as symmetry-curvature duality.

Results such as the ones by Leyton (1987) and Kimmel et al. (1995) indicate a systematic relation between symmetry axes and curvature extrema. However, the SAT has rarely been

used in practice for their identification. One recent exception is the work on part decomposition by Papanelopoulos et al. (2019), in which endpoints of the SAT are used to identify a subset of the curvature extrema and corners along object outlines, together with circular-arc outline regions where curvature is approximately constant. However, such an analysis based solely on the SAT does not capture all curvature extrema, because of its global nature, in which a part of a contour may mask an existing wavefront by interacting with another first (Belyaev and Yoshizawa, 2001).

3.8.2.4 Stability issues

One known issue of the SAT, especially in the discrete setting, is that of *stability* : the high sensitivity to noise and boundary perturbations, which often results in spurious axial branches that do not greatly contribute to the reconstruction of the input shape. A number of *significance measures* have been proposed to mitigate this issue with a procedure known as "pruning" (Shaked and Bruckstein, 1998), for example based on the propagation speed of symmetric wavefronts (Montanari, 1969; Blum, 1973; Pizer et al., 2003), based on the computation of a global minimum feature size (λ-medial axis of Chazal and Lieutier, 2005), depending on contour-based (Ogniewicz, 1992; Bai et al., 2007), branch-based (Telea, 2012), or area-based saliency measures (Shaked and Bruckstein, 1998; Leonard et al., 2016), as well as using a scale space approach in the skeletal domain (Dill et al., 1987; Ogniewicz, 1992). The choice of the method usually depends on the application, and the choice of thresholds can be challenging, similarly to the case observed for smoothing and scale spaces in the curvature domain. Shaked and Bruckstein (1998) analyse a number of significance measures developed up to 1998 under a unified framework, and observe that certain measures such as that proposed by Blum (1973) do not guarantee that symmetry axis topology is maintained. A number of SAT variants have been proposed to produce more stable symmetry axes with the aim of producing a more concise skeletal representation, closer to what could be considered the "stick-figure" of a 2D shape.

Feldman and Singh (2006) propose a Bayesian formulation of symmetry axes, viewed as generators of outline points. Kovács et al. (1998) propose an annulus — or thick maximal disk — model that maps well to psychological responses of human subjects and is by nature more robust to noise, the level of noise filtering being a function of the annulus band's width which "captures" contour points or edge data. Aparajeya and Leymarie (2016) propose an efficient implementation of this model and demonstrate its use in emphasising dominant points of articulated shapes as well as in the study of drawings and painting by famous visual artists (Leymarie and Aparajeya, 2017).

3.8.3 Perceptual grouping

Perceptual grouping is the way in which the visual system groups elements (points, segments, edges, shapes) into perceptual units , a process that is generally accepted to occur

pre-attentively (Brooks, 2015). For example, perceptual grouping is responsible for the way in which a sequence of closely spaced dots can be perceived as a single entity consisting of a curve. The identification of five fundamental perceptual grouping principles: proximity, similarity, common fate, good continuation and closure (Table 3.6), can be attributed to the pioneering works of the Gestalt school of psychology, and in particular to Wertheimer (1923). To this day, these principles are still considered to be valid and have resulted in a large body of research in the domains of perception and neuroscience as well as in a variety of computational models. The reader is referred to the excellent reviews by Wagemans et al. (2012), Brooks (2015), and Elder (2015) for comprehensive reportings on early and novel research and results on this important subject.

Principle	Short Description
Proximity	Relatively close stimuli are grouped into perceptual units
Similarity	Similar stimuli (in color/shape/orientation) are grouped into perceptual units
Common fate	Similarly moving stimuli are grouped into perceptual units
Good continuation	Tendency to group oriented elements that are perceived to be part of the same smooth curve
Closure	Tendency to complete simple shapes when these are occluded or with some gaps

Table 3.6: Main perceptual grouping principles. Refer to the chapter of Brooks (2015) for a more detailed exposition of these and a series of other more novel principles.

3.8.3.1 Contour integration

The principle of good continuation, and more specifically, the related process known as *contour integration*, are of particular interest with respect to this thesis. Contour integration is the process underlying the ability of the visual system to distinguish a curve from disjoint elements, to perceive illusory contours induced by figures such as the Kanizsa triangle or to complete contours under occlusion (Kanizsa, 1979). When considering a letterform consisting of a combination of intersecting or overlapping strokes, contour integration is likely responsible for the ability to discern the individual generating strokes rather than fuse these or mis-interpret their relationships.

Ullman (1976) poses a series of four properties that a *completion curve* must possess to connect two disjoint contour segments across an occlusion: (i) the curve should be invariant to rotation, translation and scale (*isotropy*), (ii) it should be continuous and differentiable (*smoothness*), (iii) it should minimise curvature (*minimum curvature*) and (iv) it should be invariant to a reduction of the occlusion's extent (*locality*). Kellman and Shipley (1991) propose that whether two occluded contour segments can be integrated and perceived as a unit

depends on their *relatability*, which occurs if their linear extensions intersect at an obtuse angle. Field et al. (1993) model the response of orientation-tuned cells in the primary visual cortex (V1) to adjacent cells with an *association field* that decays with distance and deviations from collinearity. A concept similar to association fields was proposed earlier by Parent and Zucker (1989) with the aim of inferring traces and curves from grayscale images. To this end, the authors propose a model that relates oriented elements based on cocircularity, i.e. whether two oriented elements are approximately tangent to the same circle, and based on a discrete partitioning into ranges of curvature magnitude. Yen and Finkel (1998) also rely on cocircularity as the basis for a biologically inspired model of V1 neurons, which results in association fields that decay with a Gaussian function of distance and deviations from cocircularity. A similar association field model is the basis of the tensor voting framework, developed by Medioni and colleagues (refer to Maggiori et al. (2015) for a technique overview), which has been widely used in pattern recognition and computer vision applications, and in particular, has been used to disambiguate overlapping parts in simple letterforms and numbers (Massad and Medioni, 2001).

A different but related approach to contour integration is pioneered by Mumford (1994), who models completion curves with a stochastic process in which tangent directions vary according to a Brownian motion. The maximum likelihood completion curve of the underlying distribution is an *elastica*, a curve than minimises the total square magnitude of curvature and is closely related to the Euler spiral (Levien, 2008). With a similar reasoning, Williams and Jacobs (1997) and later Williams and Thornber (2001) propose "stochastic completion fields", a model which explains contour integration, as well a number of illusory contour formation instances, by using particle trajectories that follow a random walk in a lattice of planar positions and orientations. Ernst et al. (2012) formulate stochastic completion fields with a generative model of the conditional link probability of one oriented element relative to another one. The probability distribution consists of the product of a radial and an angular component. The radial component is based on an exponential function that decays with distance, while the angular component parameterises deviations from perfect cocircularity and deviations from zero curvature with the product of two von Mises distributions — analogs of Gaussian distributions with circular supports. The authors empirically determine model parameters that are optimal with respect to experimental data.

3.8.4 From parts to strokes

The representations discussed in the previous sections all come into play for a representation of shape in terms of higher level perceptual units, or *parts*. Psychophysical experiments (Xu and Singh, 2002; De Winter and Wagemans, 2006), as well as results in computer vision and pattern-recognition (Siddiqi and Kimia, 1995; Macrini et al., 2008), suggest that part-like representations are intrinsic to our visual system and are essential to shape understanding

Principle	Short Description	References
Codons	Part of a contour bounded by two negative minima of curvature. Can be of 6 types.	(Richards and Hoffman, 1985)
Transversality	The union of two interpenetrating objects is likely to have a concave crease where the objects join	(Hoffman and Richards, 1984)
Minima Rule	Pairs of concavities ($m-$) are likely candidates for the segmentation of an object into parts	(Hoffman and Richards, 1984)
Limbs	Parts delimited by a line that connects two $m-$ points, and where the line forms a good continuation with the object outline at least on one side	(Siddiqi and Kimia, 1995)
Necks	Part cuts corresponding with a narrowing of the shape (local thickness minimum)	(Siddiqi and Kimia, 1995)
Short-cut Rule	The human visual system prefers to connect segmentation points that are close together. The cut (a straight line) must cross a local axis of symmetry and connect at least one negative minimum of curvature	(Hoffman and Singh, 1997)
Ligature	Centers of the SAT disks touching two distinct contour points with negative curvature	(Blum and Nagel, 1978)
Semi-ligature	Centers of the SAT disks touching one contour point with negative curvature	(Blum and Nagel, 1978)

Table 3.7: Summary of the main principles used for the decomposition of objects into parts.

and object recognition, description and categorisation (Singh and Hoffman, 2001). Consider for example the object category of "chairs with four legs" or, more specifically to the context of this thesis, the category of "X" letters consisting of two crossing strokes. Decomposing an object into perceptually meaningful parts is an ill-posed problem: multiple ambiguous hypotheses are acceptable, and their selection depends on subtle perceptual cues (De Winter and Wagemans, 2006) and on domain knowledge and functional or causal attributes (Spröte et al., 2016). However, psychophysical results suggest that similarly to perceptual grouping (Brooks, 2015), formulating early part-segmentation hypotheses (Xu and Singh, 2002) is also a low-level process that occurs pre-attentively, or at least very early in the vision process.

3.8.4.1 Part decomposition principles

A number of early theories of part decomposition assume a volumetric representation of objects, and hypothesise that parts are perceived by matching a predefined set of primitives such as generalized cylinders and cones (Marr, 1982) or "geons" (Biederman, 1987). Singh and Hoffman (2001) argue against this hypothesis, suggesting that parts emerge early in the vision process from geometric principles, "regularities of nature", which may then lead to higher level primitive based representations. Refer to Table 3.7 for a summary of a number of such principles that are dominant in the literature. Indeeed, experimental evidence suggests that part like representations are perceived rapidly (Xu and Singh, 2002), are are readily constructed from 2D silhouettes (De Winter and Wagemans, 2006), even for abstract shapes and shapes that do not have any intuitive interpretation in terms of volumetric primitives. Hoffman and Richards (1984) justify a geometric interpretation of part structure based on the "transversality principle", stating that the union of two 3D objects produces a concave crease where the two objects intersect. The projection of the silhouette for this union results in curvature minima, which, according to the so called "minima rule" (Hoffman and Richards, 1984) are the loci of a likely subdivision of an object into parts.

The minima rule provides likely candidate points for initiating a part segmentation, but it does not provide a systematic "partitioning scheme" (Siddiqi and Kimia, 1995), that determines how a shape should be decomposed. Many well-known approaches use curvature minima to define "part-lines" or "cuts" (Papanelopoulos et al., 2019), that delimit perceptually-distinct object parts. The "short-cut rule" (Singh et al., 1999) suggests that a part-line should connect one or two $m-$ points, that it should cross a local symmetry axis (in terms of Brady's SLS) and that shorter cuts and cuts connecting salient concavities (Singh and Hoffman, 2001) are preferred. Siddiqi and Kimia (1995) identify part cuts with *necks* and *limbs*. Necks coincide with a local minimum of the SAT radius, where a disk has contact with a concavity. Limbs connect concavity pairs at which the boundary has good continuation. Many part-cut based implementations exist, ranging from ones that adopt ideas from the short-cut rule (Luo et al., 2015; Wang and Lai, 2016; Papanelopoulos et al., 2019) together with additional principles such as convexity (Rosin, 2000; Liu et al., 2014; Papanelopoulos et al., 2019).

De Winter and Wagemans (2006) perform another large scale study on human preferences for segmentation points among different types of curvature extrema and inflections. The results confirm the minima rule by emphasising the perceptual importance of negative minima ($m-$), which are the most frequently chosen features by participants (approx. 64%), followed in decreasing order of importance by inflections, positive maxima ($M+$), negative maxima ($m+$), and positive minima ($M-$) and with the likelihood of a choice being correlated with the magnitude of curvature. Consistent with the minima and short-cut rules, participants show a preference for part cuts connecting two $m-$ points, followed by ones connect-

ing at least one $m-$ point. However, the short cut rule is sometimes overruled by a diagonal cut that connects a nearby $M+$ (e.g. at an "elbow"-like form).

3.8.4.2 Skeleton based methods

Already in his early works, Blum suggested that symmetry axis branches could be useful to categorise object parts, but also recognising that this part based representation may be redundant. Blum and Nagel (1978) exemplify this redundancy with the symmetry axes of a rectangle, which consists of one central axis and four smaller axes extending into corners. However, the authors also show how these somewhat redundant axes can be used to describe the rectangle having four corners, and how other SAT features characterise morphological features such as "worms", "wedges" and "flexures".

Semi-ligatures and *ligatures* are symmetry axis segments where the maximal disks touch the contour at points that are part of respectively one or two distinct curvature minima or corners (Blum and Nagel, 1978; August et al., 1999). These tend to be symmetry axis segments that contribute to relatively small portions of the outline, but can act as "glue" that connects segments of perceptually distinct outline parts (De Winter and Wagemans, 2006; Macrini et al., 2008). Macrini et al. (2008) exploit ligatures and semi-ligatures to abstract and decompose the SAT into a graph that describes parent-child relations among object parts called "bones". The method uses ligature analysis to categorise branching points into *junctions*, which are classified as one of: *Y-junctions*, *P-junctions* (where "P" stands for protrusion), and *nested-junctions* (combinations of the previous two).

Rom and Medioni (1993) use axes of the SLS together with their cross sections (*ribbons*), to decompose an outline into parts. The SLS is computed from a B-spline approximation of the outline and the method also uses SLS axes to identify morphological features such as "terminations" and "bends" as well as topological features such as "mushrooms" ("T"-like features). Mi and DeCarlo (2007) also use the SLS to compute parts, and identify hyperbolic regions along the outline that determine the decomposition of an object into parts and transition regions where parts smoothly blend. Feldman and Singh (2006) propose a Bayesian interpretation of the skeleton, aimed at producing a more intuitive decomposition into parts than the one produced by the SAT. Shape is seen as a stochastic process that grows the outline from the skeleton, and the skeleton is given by the maximum a posteriori (MAP) estimate of the process parameters. Froyen et al. (2015) implement this concept by estimating the parameters of a mixture of splines paired with Gaussian thickness profiles with a variant of Bayesian Hierarchical Clustering (Heller and Ghahramani, 2005).

3.8.4.3 Overlapping parts and vectorisation

Very few methods consider the problem of potentially *overlapping parts*, which also pertains to the tasks of decomposing glyphs or drawings into constituent strokes. With the aim of vectorisation, Luo et al. (2015) and more recently Kim et al. (2018) propose a data-driven method

that can vectorise overlapping parts of Chinese characters. The method of Froyen et al. (2015) can disambiguate overlapping parts, but it is demonstrated only on relatively simple tubular objects. Favreau et al. (2016) propose another approach that uses a Monte-Carlo exploration method to create vectorisations of thin line drawings that maximise a tradeoff between simplicity and reconstruction accuracy. The problem of disentangling potentially overlapping parts also relates to *multi-manifold learning* (Arias-Castro et al., 2017; Goldberg et al., 2009; Deutsch and Medioni, 2017), which is the segmentation of data samples generated by multiple, potentially-intersecting manifolds. Massad and Medioni (2001) model contour integration with tensor voting and use the resulting tensor fields to identify *junctions* where parts of an objects cross and overlap and to compute occluded contours. The authors demonstrate results on simple letterforms.

3.9 Summary

To summarise, we have exposed a number of topics starting from curve generation and stylisation methods, going into a number of notions from the study of human movement, followed by a review of existing approaches for the representation, stylisation and generation of stylised strokes, typography, calligraphy and handwriting, and finishing up with a review of shape representations, with a focus on ones that have a perceptual grounding.

The emphasis on human movement in Section 3.5 serves as a background in order to explore the hypothesis that a *movement centric* and *process based* representation of shape is fundamental for the study and procedural generation of graffiti, but more in general for the study of any other art form. Quoting De Preester (2013, pg. 21):

> "How a musical passage is played, how a monologue is delivered, how a piece of fruit, a tree, or a person is delineated and shaded on canvas, how a dance ensemble spreads apart and gathers together – all such artistic realities depend on the living movement dynamics of the artists creating or performing the work...".

Surprisingly, both in Western art-theoretical literature (Fong, 2003) as well as in the computer graphics domain (Kyprianidis et al., 2013), the notion of movement in the study of artistic styles is seldom taken into account. While dynamic models of movement are central in the handwriting-synthesis and graphonomics domains (Section 3.5.3), such representations are rarely used in more art/style centric generative applications, with the exceptions of a number of methods aimed at generating synthetic calligraphy (Shinoda et al., 2003; Wang et al., 2005; Fujioka et al., 2006; Fujioka and Miyata, 2011) or mimicking hand-drawm images (AlMeraj et al., 2009; House and Singh, 2007). In Chapter 4 and Chapter 5 we will develop two calligraphic stylisation methods that systematically take into account a number of the principles discussed in Section 3.5 to produce trajectories that resemble the ones that would be made by an expert graffiti artist. These methods are also aimed at addressing a number of

limitations that we have encountered in Section 3.3, when it comes to specifying and editing calligraphic curves with geometry-based curve manipulation methods.

In Section 3.6 we have seen some useful letterform descriptors with a focus on stroke-based representations. These descriptors are the basis for the representation that will be used in Chapter 10 to recover strokes and their connectivity relations from gylph outlines. The emphasis on strokes is based on commonly held notions in typography and calligraphy (Noordzij, 2005; Wang, 2013), but also on results (Section 3.6.2, Section 3.7.2) that demonstrate the power and flexibility of this representation and finally motivated by its compatibility nwith the proposed movement centric approach to curve generation. We have concluded that there is no stroke representation in the literature that robustly supports effects that are typically seen in graffiti pieces, such as self overlaps and local layering. We will address this gap in Chapter 6 with a variant of skeletal strokes specifically design to reproduce graffiti and compatible with a movement centric approach to curve generation. A survey of font,calligraphy and handwriting generation methods (Section 3.7) reveals that many methods rely on recovering stroke-based representations from font outlines (Table 3.4), but none of these methods is general enough to operate on fonts with arbitary languages, styles or writing systems.

The emphasis on shape in Section 3.8 serves as a background for the development of a solution to this gap in Chapter 10, and more in general as a foundation for all the methods developed in Part II of the thesis. A number of these topics, in particular the ideas of Michael Leyton (Leyton, 1987, 1988) and an understanding of symmetry based shape descriptors, will inform the development of curvilinear shape features in Chapter 7, a shape representation that we will use to recover movement primitives (Chapter 8) and strokes (Chapter 10) from existing geometry. The first procedure will serve as a basis to develop, in Chapter 9, a kinematics-based analogue of an example-based approach to curve stylisation that is popular in the literature (Hertzmann et al., 2002; Li et al., 2013; Lang and Alexa, 2015). The second procedure will allow us to recover letter structures from fonts, which ultimately guides the proposed graffiti generation and stylisation framework (Chapter 11).

Part I

Part I - Kinematic and geometric primitives for interactive graffiti art generation

Chapter 4

Calligraphic stylisation: the Sigma-Lognormal model

This chapter is based on work initially developed independently and resulting in two publications (Berio et al., 2016; Berio and Leymarie, 2015), and then in collaboration and with the additional advice of Professor Réjean Plamondon at Polytechnique Montréal (Berio et al., 2017d,a, 2018b,a). .

In the next two chapters, we will explore two methods that instantiate the previously discussed ideas of kinematic curve design and *calligraphic stylisation*. In this chapter, we focus on the field of *graphonomics* (Kao et al., 1986) and in particular on the Sigma Lognormal ($\Sigma\Lambda$) model. On the basis of the Kinematic Theory (Plamondon, 1995), the $\Sigma\Lambda$ model (Leiva et al., 2017) is a physiologically plausible model of handwriting which describes the kinematics of arbitrarily complex pen movements through the superposition of a sequence of target directed (ballistic) sub-movements, which are characterised by a *lognormal* speed profile.

Plamondon suggests that the theory is aimed at describing the motions of "humans that are in perfect control of their movements" (Plamondon et al., 2013), which he refers to as the *lognormality principle*, where with experience/practice, the velocity profile of a hand movement tends to converges towards a sum of lognormal functions. This concept fits well with the notion that tagging/drawing movements and forms of graffiti tags and calligraphy result from extensive years of practice, and with the hypothesis that the experienced artist, after years of practice, will be capable of synthesising effortlessly (without thinking) aesthetically pleasing and distinct traces. As a result, graffiti can be seen in the context of the Kinematic Theory as one possible artistic instantiation of the lognormality principle.

From a motor control perspective, lognormals have been experimentally shown to be accurate descriptors of human movement speed profiles (Rohrer and Hogan, 2006), and the

ΣΛ model produces kinematics that are similar to the ones that would be produced by a human while drawing or writing. The model is originally conceived for the analysis and synthesis of handwriting for pattern-recognition and biometric purposes. However, we will show how, through an appropriate re-parameterisation, such a framework can be used in similitude to established popular spline-based methods, with the additional benefit of capturing both the geometry and dynamics of a human made trace with a single integrated representation. This results in a system where trajectories that resemble the ones made with a freehand movement are specified and edited with a sparse motor plan, rather than a sketch-based input and interface.

The following sections describe three variations of the Sigma Lognormal (ΣΛ) model (Plamondon et al., 2014) developed in the context of this thesis: the ΣΛ model in its original formulation (Section 4.1) followed by two extensions: the Weighted ΣΛ (Section 4.2.1) and the Spiral ΣΛ (Section 4.2.2) models. These two extensions are particularly aimed at interactive CAGD applications, and their uses and advantages for the interactive generation (Section 4.3), variation (Section 4.4) and rendering (Section 4.5) of strokes are discussed last.

4.1 Sigma Lognormal Model

The Kinematic Theory of Rapid Human Movements (Plamondon, 1995), together with its derived models, abstracts the complexity of the neural and muscular processes underlying human movement formation with a "black box" model, consisting of a large number of hierarchically coupled linear sub-systems. Plamondon et al. (2003) show that as the number of sub-systems grows, the impulse response of such a system to a centrally generated command converges to a *lognormal* (eq. 4.1), which accurately describes the variably asymmetric bell shape that commonly characterises the velocity of human target directed movements (Plamondon et al., 1993; Rohrer and Hogan, 2003). The velocity of the resulting movement is modelled with a time shifted lognormal function: [1]

$$\Lambda(t) = \frac{1}{\sigma\sqrt{2\pi}(t-t_0)} \exp\left(-\frac{(\ln(t-t_0)-\mu)^2}{2\sigma^2}\right) \quad , \tag{4.1}$$

where t_0 is the *activation time* of a centrally generated command (e.g. by the central nervous system or CNS), and the parameters μ and σ model the delay and response time of the system to the command in a natural-logarithmic time scale, while also determining the asymmetry and support of the lognormal.

With the specific objective of modeling handwriting movements, the ΣΛ model describes arbitrarily complex movement trajectories with the space-time superposition of a discrete number of lognormal sub-movements. Each such sub-movement is aimed at an

[1] In statistics, this is more commonly referred to as the 3 parameter lognormal. Note that in the Kinematic Theory, the function describes an impulse response, not a probability density function.

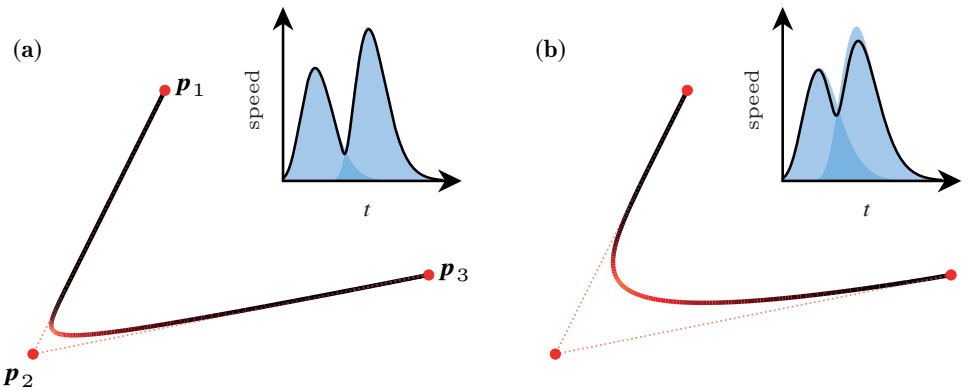


Figure 4.1: The effect of different time overlaps for two lognormals. The activation of the second lognormal in **(b)** is anticipated with respect to **(a)**. The resulting trajectories are displayed in black, with a red shade indicating the region in which the cumulative influence of both submovements is highest. The red circles are the initial position ($\boldsymbol{p}_0$) and two virtual targets ($\boldsymbol{p}_1, \boldsymbol{p}_2$). Note that these positions define a motor plan (dashed red).

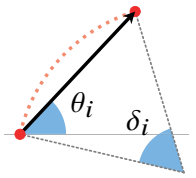
imaginary location referred to as *virtual target*. The velocity of each sub-movement is determined by a lognormal $\Lambda(t)_i$, which is computed according to Eqn. 4.1 and with t_0, μ_i, σ_i its activation time, delay and response time. Intuitively, initiating a lognormal $\Lambda(t)_i$ for one sub-movement while the lognormal $\Lambda(t)_{i-1}$ for another sub-movement is still being executed, results in a superposition that produces a smooth trajectory that combines the two. Anticipating the activation time for $\Lambda(t)_{i-1}$ increases the time overlap between lognormals and results in a smoother trajectory (Figure 4.1).

With the assumption that handwriting movements are executed with rotations of the elbow or wrist, the $\Sigma\Lambda$ model describes the geometry of a sub-movement with an oriented circular arc, the curvilinear evolution of which is computed from the integral of equation (4.1):

$$w_i(t) = \int_0^t \Lambda_i(u)\,du = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{\log(t - t_{0i}) - \mu_i}{\sigma_i\sqrt{2}}\right)\right] \quad , \tag{4.2}$$

giving the curvilinear evolution function

$$\phi_i(t) = \theta_i - \frac{\delta_i}{2} + \delta_i\, w(t) \quad , \tag{4.3}$$



such that θ_i is an *orientation* parameters that determines the angle that a sub-movement makes with the horizontal axis and δ_i is a *curvature* parameter that determines the internal angle of the assumed circular arc (see inset on the left for an illustration). The planar pen-tip velocity for a trajectory is then calculated with the vectorial superposition of M sub-movements:

$$\dot{\boldsymbol{x}} = \sum_{i=1}^{M} D_i \Lambda_i(t) \begin{bmatrix} \cos(\phi_i(t)) \\ \sin(\phi_i(t)) \end{bmatrix} \quad , \tag{4.4}$$

where D_i is an *amplitude* determining the distance covered by each sub-movement. A planar trajectory can be generated by integrating equation (4.4) starting from an initial position $\boldsymbol{p}_o$ with:

$$\boldsymbol{x}(t) = \boldsymbol{p}_0 + \int_0^t \dot{\boldsymbol{x}}(u)\,du \tag{4.5}$$

Displacement based solution. Exploiting the closed form integral in equation (4.2) allows to efficiently compute the trajectory in terms of a sum of displacements $\boldsymbol{s}_i(t)$, with:

$$\boldsymbol{x}(t) = \boldsymbol{p}_0 + \int_0^t \dot{\boldsymbol{x}}\,du = \boldsymbol{p}_0 + \sum_{i=1}^{M} \int_0^t D_i \frac{d}{du} w_i(u) \begin{bmatrix} \cos(\phi_i(u)) \\ \sin(\phi_i(u)) \end{bmatrix} du = \boldsymbol{p}_0 + \sum_{i=1}^{M} \boldsymbol{s}_i(t)\,,$$

where

$$\boldsymbol{s}_i(t) = \frac{D_i}{\delta_i} \begin{bmatrix} \left(\sin\left(\phi_i(t)\right) - \sin\left(\theta_i - \delta_i/2\right)\right) \\ \left(\cos\left(\phi_i(t)\right) - \cos\left(\theta_i - \delta_i/2\right)\right) \end{bmatrix} \qquad \text{if } |\delta_i| > 0\,,$$

$$\text{and} \quad \boldsymbol{s}_i(t) = D_i \begin{bmatrix} w_i(t)\cos\theta_i \\ w_i(t)\sin\theta_i \end{bmatrix} \qquad \text{otherwise.}$$

This allows to efficiently compute the pen tip position at a given time t in parametric form by exploiting the error function (erf), which is implemented in most programming languages and numerical packages and avoids the need for numerical integration.[2]

[2] This parameterisation was derived independently in the context of this thesis, during the development of the weighted ΣΛ model. However, it has also been previously proposed by O'Reilly and Plamondon (2009) so it is now described as part of the original model formulation.

Curvature. The acceleration components of the lognormal trajectory are then given by (Plamondon and Guerfali, 1998a):

$$\ddot{x} = \sum_{i=1}^{N} D_i \dot{\Lambda}_i(t) \cos(\phi_i(t)) - D_i \delta_i \Lambda_i^2(t) \sin(\phi_i(t)) \quad , \tag{4.6}$$

$$\ddot{y} = \sum_{i=1}^{N} D_i \dot{\Lambda}_i(t) \sin(\phi_i(t)) + D_i \delta_i \Lambda_i^2(t) \cos(\phi_i(t)) \quad , \tag{4.7}$$

with

$$\dot{\Lambda}_i(t) = \Lambda_i(t) \frac{\mu_i - \sigma_i^2 - \log(t - t_{0i})}{\sigma^2 (t - t_{0i})} \quad , \tag{4.8}$$

which allows us to compute the curvature function at time t with Eqn. 2.1.

4.2 ΣΛ model for calligraphic stylisation

The conventional ΣΛ model definition above, implicitly defines a motor plan P consisting of an initial position $\boldsymbol{p}_0$, followed by M targets $\boldsymbol{p}_1 \ldots \boldsymbol{p}_M$ at which consecutive sub-movements are aimed. However such a definition suffers from poor locality and is not well suited for our tasks of interactive editing and calligraphic stylisation. As an example, changing a curvature parameter δ_i for one sub-movement, modifies the location of all consecutive virtual targets and the corresponding trajectory segments. For our use case, we seek instead a parameterisation that clearly separates the definition of a motor plan from a set of parameters that determine its kinematic realisations. To this end, we develop two variants of the ΣΛmodel: The *weighted Sigma Lognormal* ($\omega\Sigma\Lambda$) *model* and the *weighted Euler spiral Sigma Lognormal* ($\omega\mathscr{E}\Sigma\Lambda$) *model*. The first variant is a simple reformulation of the model with an explicitly defined motor plan. The second variant, is an extension of the first that allows for more complex sub-movement primitives than circular arcs. Finally, we describe an intermediate parameterisation of t_{0i}, μ_i, σ_i that is specifically aimed at the stylisation and interaction use cases.

4.2.1 The weighted Sigma Lognormal ($\omega\Sigma\Lambda$) model

We define a *weighted* parameterisation of the model by computing the amplitude and orientation parameters D_i and θ_i from an explicitly defined motor plan with vertices $\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_M$, which defines the initial position $\boldsymbol{p}_0$ followed by M virtual targets (Figure 4.2).

The parameters θ_i are given by the orientations of the vectors $\boldsymbol{p}_i - \boldsymbol{p}_{i-1}$ connecting consecutive targets. The sub-movement amplitudes are given by

$$D_i = \begin{cases} \frac{\delta_i \| \boldsymbol{p}_i - \boldsymbol{p}_{i-1} \|}{2 \sin(\delta_i / 2)} & \text{if } |\delta_i| > 0 \quad , \\ \| \boldsymbol{p}_i - \boldsymbol{p}_{i-1} \| & \text{otherwise} \quad , \end{cases} \tag{4.9}$$

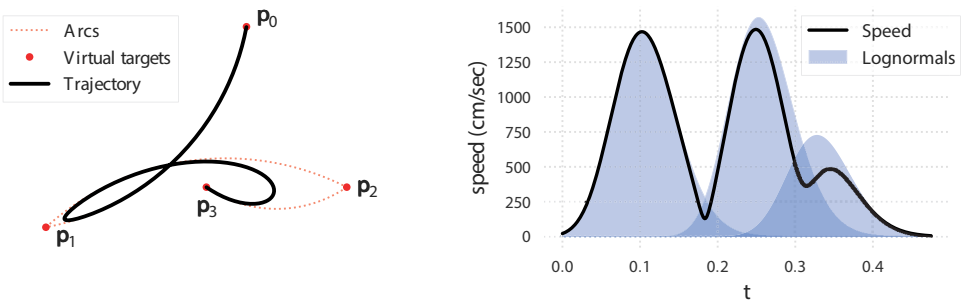


Figure 4.2: Left: ΣΛ trajectory (in black) with the corresponding motor plan visualised as targets (red dots) connected by circular arcs (dashed red) rather than straight line segments.. Right: The corresponding speed profile with different time overlaps between lognormals.

which adjusts the command amplitude depending on the distance to a virtual target and on the ratio between the perimeter and the chord length of the corresponding circular arc .

This reparameterisation results in a clear separation between the motor plan P and a set of kinematic parameters (the remaining ΣΛ parameters), which determine the fine evolution of a trajectory that follows the motor plan, that is its kinematic realisation $\mathscr{P}$. Note that for the case of the ΣΛ and ωΣΛ models, we will visualise motor plans with vertices connected by circular arcs, rather than straight line segments (Figure 4.2). This is done with the purpose of visualising the geometry of each sub-movement, while avoiding clutter in the figures. However, the motor plan remains a sequence of vertices connected by polylines and the kinematic parameters δ_i that determine the arc geometry should not be considered part of the motor plan's definition. In the next section, we will use a similar visualisation approach for sub-movements consisting of geometric primitives other than circular arcs.

4.2.2 The Weighted Euler Spiral Sigma Lognormal ($\omega\mathscr{E}\Sigma\Lambda$) Model

The ΣΛ formulation is flexible enough to accommodate for movement primitives with geometries other than straight or circular arcs. With the use case of interactive curve editing, we can extend the weighted ΣΛ model with primitives consisting of Euler spiral segments.

Euler spirals (Levien, 2008) (a.k.a. Cornu spirals, or clothoids or spiros) are curves in which curvature varies linearly with arc length, permitting the description of variably curved segments that may contain an inflection. At the expense of adding a supplementary parameter per sub-movement to the model, the use of Euler spirals reduces the number of virtual targets needed to define a trajectory — such as when defining a doubly looping eight ("8") (Berio and Leymarie, 2015) — and provides an additional level of editing flexibility. The resulting method can also be used to define trajectories that are identical to the standard ΣΛ model, since in the limit an Euler spiral segment converges to a circular arc (Walton and

Meek, 2008).

The coordinates of an Euler spiral for a given arc length parameter s can be retrieved with the cosine ($C(s)$) and sine ($S(s)$) Fresnel integrals (Levien, 2009a) :

$$C(s) = \int_0^s \cos\left(u^2\right) du \quad \text{and} \quad S(s) = \int_0^s \sin\left(u^2\right) du \quad , \tag{4.10}$$

which can be efficiently approximated with a numerical method described by Heald (1985).

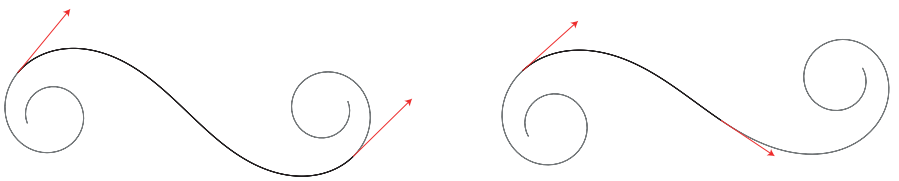


Figure 4.3: Examples of Euler spiral sub-movements using different Hermite constraints (in red).

The curvilinear evolution of each sub-movement is determined by *two* arc length values s_{0i} and s_{1i} which, although they are not intuitive to grasp, can be uniquely computed given the orientation of two tangents (a.k.a. Hermite constraints) with respect to the chord of the spiral (Figure 4.3). A number of methods exist for this task (Kimia et al., 2003; Walton and Meek, 2008; Levien, 2009a; Bertolazzi and Frego, 2013); in this thesis we use the secant method proposed by Levien (2009a), which has experimentally proven to be fast and robust. We then define the arc length evolution of the spiral for each sub-movement with:

$$\phi_{si}(t) = s_{0i} + (s_{1i} - s_{0i})\, w_i(t) \tag{4.11}$$

and the integrated displacement of each sub-movement with:

$$\boldsymbol{d}_i(t) = \frac{D_i}{l_i} \begin{bmatrix} (C(\phi_{si}(t)) - C(s_{0i}))\cos(\theta_i - \theta_{ci}) - S(h_i\phi_{si}(t)) - S(h_i s_{0i}))\cos(\theta_i - \theta_{ci}) \\ (C(\phi_{si}(t)) - C(s_{0i}))\sin(\theta_i - \theta_{ci}) + S(h_i\phi_{si}(t)) - S(h_i s_{0i}))\cos(\theta_i - \theta_{ci}) \end{bmatrix} , \tag{4.12}$$

where

$$l_i = \sqrt{(C(s_{1i}) - C(s_{0i}))^2 + (S(s_{1i}) - S(s_{0i}))^2} \quad \text{and} \quad \theta_{ci} = \tan^{-1}\left(\frac{S(s_{1i}) - S(s_{0i})}{C(s_{1i}) - C(s_{0i})}\right) \tag{4.13}$$

are used to respectively correct for the chord length and orientation of the spiral in its canonical form, and where

$$h_i = \operatorname{sgn}\left(\frac{s_{1i}^3}{2|s_{1i}|} - \frac{s_{0i}^3}{2|s_{0i}|}\right) \tag{4.14}$$

takes care of flipping the spiral along the horizontal axis depending on its curvature and thus

allows to capture different combinations of user defined Hermite constraints.

4.2.3 Lognormal timing reparameterisations

While the lognormal function works remarkably well in describing the form of human movement speed profiles (Rohrer and Hogan, 2006), its parameters do not have an intuitive correlation with its mode and shape (Mandelbrot, 1997). For example, with the standard lognormal parameterisation, the variation of μ shifts the onset time of the lognormal (Figure 4.4).

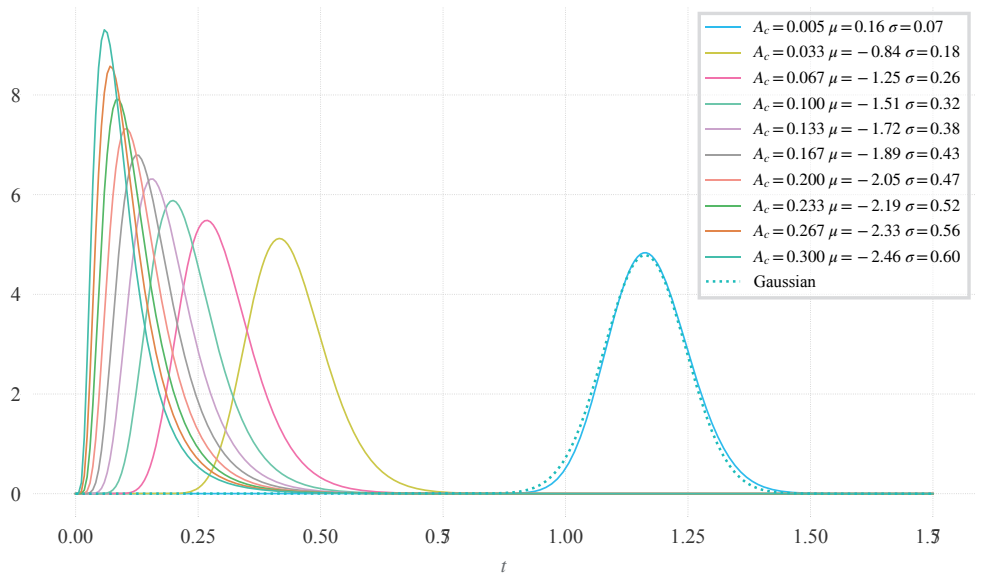


Figure 4.4: Lognormals for different values of A_{c_i} with $T_i = 0$. As $A_{c_i} \to 0$ the lognormal converges to a Gaussian. An example Gaussian centered at the mode of the lognormal with $Ac = 0.005$ is shown in dashed blue for comparison.

A precise specification of timing and profile shape of each sub-movement can be facilitated with an intermediate parameterisation that takes advantage of a few known properties of the lognormal (Djioua and Plamondon, 2008b). Each sub-movement can be reparameterised with: (i) a sub-movement duration T_i, (ii) a relative time offset Δt_i with respect to the previous sub-movement time occurrence and duration, and (iii) a shape parameter A_{c_i} $\in (0,1)$, which defines the skewedness of the lognormal (Plamondon et al., 2003). The $\Sigma\Lambda$ parameters $\{\mu_i, \sigma_i\}$ can be then computed with:

$$\sigma_i = \sqrt{-\log(1 - A_{c_i})}, \qquad \mu_i = 3\sigma_i - \log(\frac{-1 + e^{6\sigma_i}}{T_i}) \tag{4.15}$$

As A_{c_i} approaches 0, the shape of the lognormal converges to a Gaussian (Figure 4.4), with mean $t_{0i} + e^{\mu_i - \sigma_i^2}$ (the mode of the corresponding lognormal) and standard deviation $\frac{T_i}{6}$.

The activation times are computed from the relative time offsets and are given by:

$$t_{0i} = t_{0i-1} + \Delta t_i \sinh(3\sigma_i) \quad , \tag{4.16}$$

with $i > 1$ and $t_{01} = 0$. The parameter Δt_i then intuitively determines the smoothness of the trajectory similarly to weights in NURBS curves; smaller values increase the lognormal overlap and consequently produce a smoother trajectory in the vicinity of the virtual target (Figure 4.1).

Notes on the biological interpretation of the $\Sigma\Lambda$ ***parameters.*** The proposed parameterisation is convenient for the generative or interactive specification of trajectories, in which it may be useful to precisely define the time occurrence and the duration of a sub-movement. However, this comes at the cost of a biologically plausible interpretation of the parameters. First, we explicitly determine the duration of a sub-movement (T_i) which is biologically questionable (Harris and Wolpert, 1998). Second, an examination of Figure 4.4 raises questions on the interpretation of the parameter t_{0i} as the activation time of a sub-movement at the CNS level. In fact, as $A_{c_i} \to 0$ and the lognormal becomes more symmetric, its mode[3] and onset tend to infinity. At the same time, it is known to be possible for the speed profiles of human movements to assume a nearly symmetric or negatively skewed shape (Flash and Hogan, 1985; Nagasaki, 1989; Engelbrecht, 2001).

One method to address the symmetry issue is to use a different, support-bounded formulation of the lognormal, which has been previously proposed by Plamondon (1993) as an alternative to Eqn. 4.1. This formulation requires an additional parameter, but also permits negatively skewed or symmetric speed profiles without incurring the previous time delay issue. However, the parameterisation also assumes a predetermined movement duration. Another approach is to use the more sophisticated Delta Lognormal model, which describes each aiming (sub)movement with the sum of one agonist and one antagonist lognormal, and also enables both negatively skewed or nearly symmetric speed profiles. While these are interesting avenues of future research, the issues raised above can be considered negligible when considering our use case, which requires generating biologically plausible kinematics but not necessarily an accurate modelisation of all the main features of the human trajectory formation process.

4.3 User interaction

The $\omega\Sigma\Lambda$ and $\omega\mathscr{E}\Sigma\Lambda$ models provide flexible trajectory generation tools that are particularly well suited for interactive (point and click) editing procedures. For example, the user can

[3] Note that the peak of the lognormal is located at the mode, rather than at the mean.

easily edit the spatial evolution of the trajectory by dragging virtual targets, and modify the sub-movement shapes and timing properties by manipulating handles placed in correspondence with motor plan vertices.

The resulting user interface (UI) is very similar to the ones used in traditional methods such as based on Bézier curves. However, the proposed method also facilitates the *dynamic* production of curves used in art forms such as calligraphy or graffiti. Targets are located in proximity of curvature extrema along the generated trajectory, which are known to be highly informative (Attneave, 1954) and perceptually salient (De Winter and Wagemans, 2008b), and prove good candidates for the interactive definition of curves (Levien and Séquin, 2009; Yan et al., 2017). At the same time the user is effectively editing a plan for an intended motion with a representation that reflects the concatenation of a series of simple reaching/aiming movements. Such a target mapping is consistent with the hypothesis of an abstract end effector independent representation of movements in the brain (Ferrer et al., 2015).

Circular arc interaction. For the case of the $\omega\Sigma\Lambda$ model, all targets with the exception of the last are associated with a handle originating at the target locus. The angle between the handle and the segment connecting the consecutive target is $\delta_i/2$ and determines the arc internal angle. The length of the handle is inversely proportional to Δt_i and determines overlap between lognormals. A longer handle results in a smaller value of Δt_i and in an accordingly smoother trajectory in the vicinity of the target (Figure 4.5a). A user can click to create a new target point, resulting in a new initially straight sub-movement with default values of $\delta_i = 0$ and $\Delta t_i = 0.5$, resulting in an average overlap between adjacent sub-movements.

Euler spiral interaction. For the case of the $\omega\mathscr{E}\Sigma\Lambda$, we add two handles for each motor plan segment, originating at each segment endpoint (Figure 4.5b). The orientations of the handles with respect to the segment determine the tangents that are used to compute the Euler spiral parameters s_{0i}, s_{1i} with the method by Levien (2009b). The length of the first handle determines the time overlap parameter Δt_i similarly to the circular arc case.

4.4 Kinematic variability and stylisation

The $\Sigma\Lambda$ model directly reflects the characteristics of a smooth human movement at the planning (targets and motor plan) and neuromotor level (the remaining parameters). We therefore expect and observe that parameter perturbations result in variations of a trace that are similar to the one that would be seen in multiple instances of handwriting or drawing made by one or more subjects (Figure 4.6). The variability produced by the $\Sigma\Lambda$ model is not a by-product of a set of instances, computed afterward, but is rather intrinsically built in the abstract representation of a pattern. In previous works, this property of the $\Sigma\Lambda$ model has been exploited to produce artificial data for handwriting recognisers (Fischer et al., 2014), signature verifiers (Galbally et al., 2012; Diaz-Cabrera et al., 2018), gesture graphical input (Leiva et al., 2016, 2017). As we shall see the same property can also be exploited for the more artis-

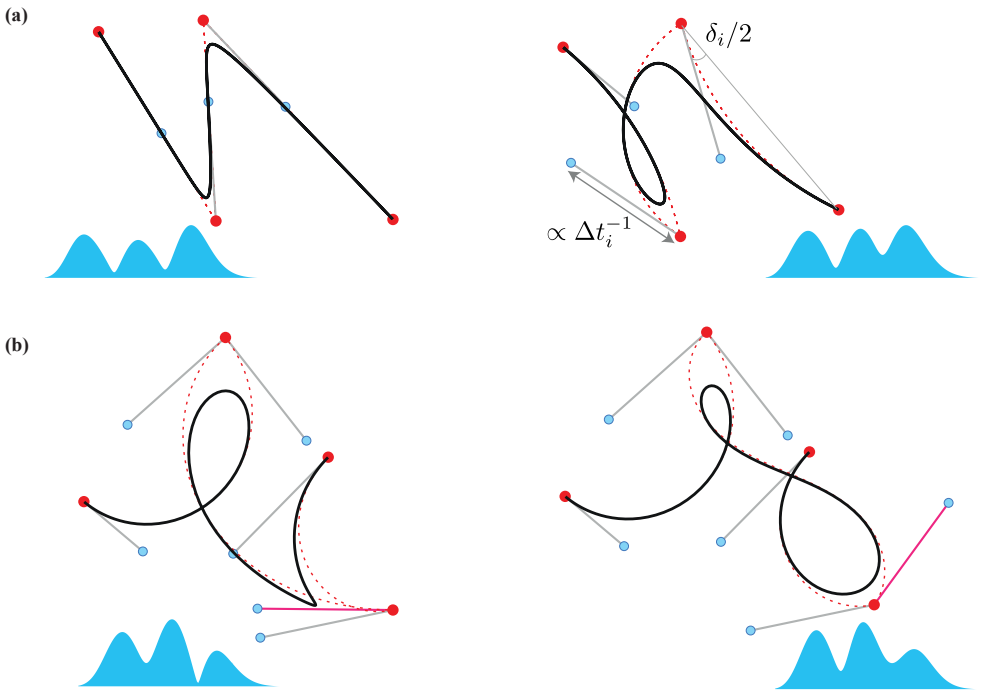


Figure 4.5: Example UI for editing ΣΛ trajectories, with speed profiles for each trajectory shown in cyan below. **(a)** $\omega\Sigma\Lambda$ model with circular arc primitives. *Left*: default configuration when user adds new targets. *Right*: trajectory after some manipulations. Each motor plan vertex, with the exception of the last, has a handle (grey segment terminating in a blue dot) that can be dragged to control the values of the sub-movement time overlap and curvature parameters Δt_i and δ_i.The length of the handles (defined with a blue dot) is inversely proportional to the value of Δt_i and the angle of the handle with respect to the vector between two consecutive targets is $= \delta_i/2$. **(b)** $\omega\mathscr{E}\Sigma\Lambda$ model with Euler spiral primitives. In this case, the geometry of each primitive is determined by *two* handles originating at the endpoints of each motor plan segment. The spiral parameters are determined by the angles made by the handles with the motor plan segment. The examples show how rotating the handle (emphasised in red) transforms the trajectory on the left into the one on the right, resulting in a sub-movement that contains an inflection.

tically oriented procedural generation and stylisation applications.

4.4.1 Artificial variability

The proposed intermediate ΣΛ parameterisation is useful in an interactive setting, but it also facilitates the generation of artificial variations of an input trajectory. In fact, applying perturbations at the level of the parameters Δt_i and δ_i and to the explicitly defined target positions $\boldsymbol{p}_i$, avoids issues with error propagation when considering the model in its original formulation (Plamondon et al., 2014).

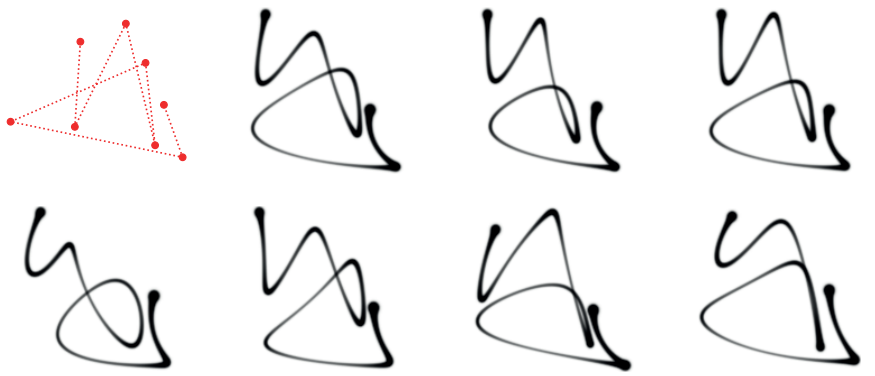


Figure 4.6: Target structure of a letter "a" (top left) and kinematic variations of its trace generated by perturbing ΣΛ parameters.

In our experiments, when perturbing targets, we have found that applying the perturbation with a variance inversely proportional to the temporal overlap parameters Δt_i improves the legibility of the variations (Figure 4.6). This corresponds to imposing a higher precision requirement at trajectory locations with higher curvature, locations that are known to be the most informative of a trace/contour (Feldman and Singh, 2005). From a motor control perspective, this in effect is consistent with the "minimal intervention principle" (Todorov, 2004), suggesting that human movement variability is higher where it does not interfere with the performance required for a task.

More specifically, we adjust the virtual targets with:

$$\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i + \Delta t_i^{-1} \boldsymbol{\epsilon}_p$$

where $\boldsymbol{\epsilon}_p \sim \mathcal{N}\left(\mathbf{0}, \sigma_p \bar{s} \boldsymbol{I}\right)$ is normally distributed with σ_p a user configurable variance , $\bar{s}$ is the average distance between virtual targets and Δt_i^{-1} modulates the perturbation so it is inversely proportional to the time offset parameter Δt_i. The kinematic parameters δ_i and Δt_i are perturbed with

$$\delta_i \leftarrow \delta_i \epsilon_\delta \qquad \text{and} \qquad \Delta t_i \leftarrow \Delta t_i \epsilon_\Delta$$

where $\epsilon_\delta \sim \mathcal{N}(1, \sigma_\delta)$, $\epsilon_\Delta \sim \mathcal{N}(1, \sigma_{\Delta t})$ and where σ_δ and $\sigma_{\Delta t}$ determine the variance of δ_i and Δt_i respectively.

4.4.2 Stylistic variations

By construction, the ΣΛ model with the proposed weighted parameterisation instantiates a bi-level representation that is compatible with the previously introduced concept of "style as kinematics". The target positions describe a motor plan, providing a sparse structural descriptor of a family of traces (Figure 4.7a). The remaining ΣΛ parameters determine the parameter space describing this family of traces (Figure 4.7b). While a user can always adjust

the parameters of one or more ΣΛ primitives interactively, for the task of stylisation we seek to produce stylistic variations over a given motor plan that are applied consistently across one or more trajectories.

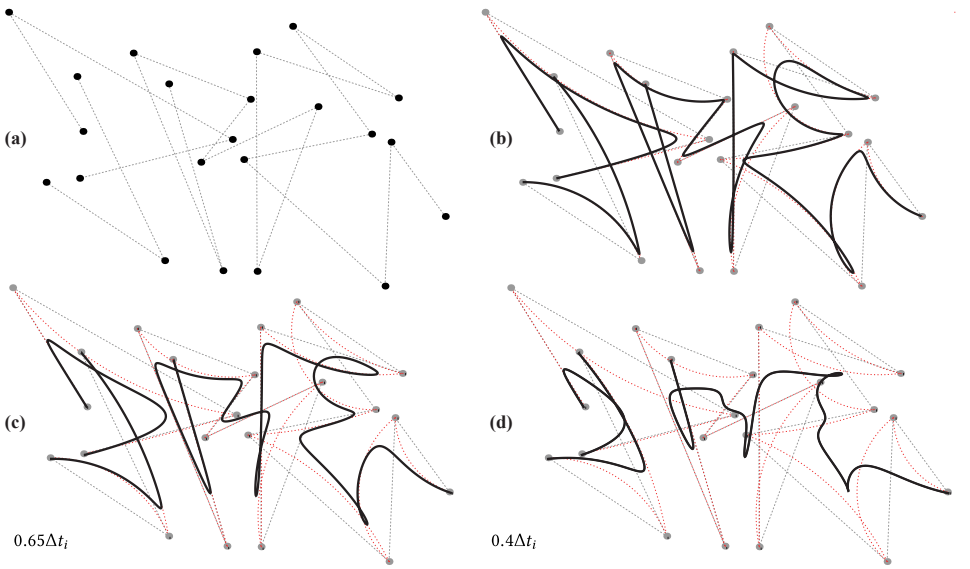


Figure 4.7: **(a)** Motor plan for a tag "PRE" and **(b)** a trajectory generated with user defined ΣΛ parameters. **(c)** Globally scaling all Δt_i parameters by a factor of 0.65 results in a smoothing effect. **(d)** A lower scaling factor of 0.4 results in most of the recognizable structure of the pattern being lost.

A trivial method to generate stylistic variations is to simply scale the Δt_i parameters (Figure 4.7c). However, this quickly produces a degradation of the trace (similarly to the smoothing effect of a convolution) and for certain instances of letterforms will result in a loss of structure and legibility (Figure 4.7d).

4.4.2.1 Key-point adjustment

One method to overcome this limitation is to adjust the motor plan so the trajectory follows the structure of the original motor plan more closely, even when lognormal primitives have a large degree of overlap. To do so, we identify a series of $M-1$ *key-points* $\{\tau_i\}_{i=1}^{M-1}$, locations that approximately correspond to curvature extrema along the generated trajectory and indicating the time occurrence at which the influence of one lognormal exceeds the previous one (Figure 4.8a). The time occurrence of each key-point is computed at the intersection of the scaled profiles of consecutive primitive pairs by solving:

$$D_i\Lambda(t)_i - D_{i+1}\Lambda(t)_{i+1} = 0 \quad ,$$

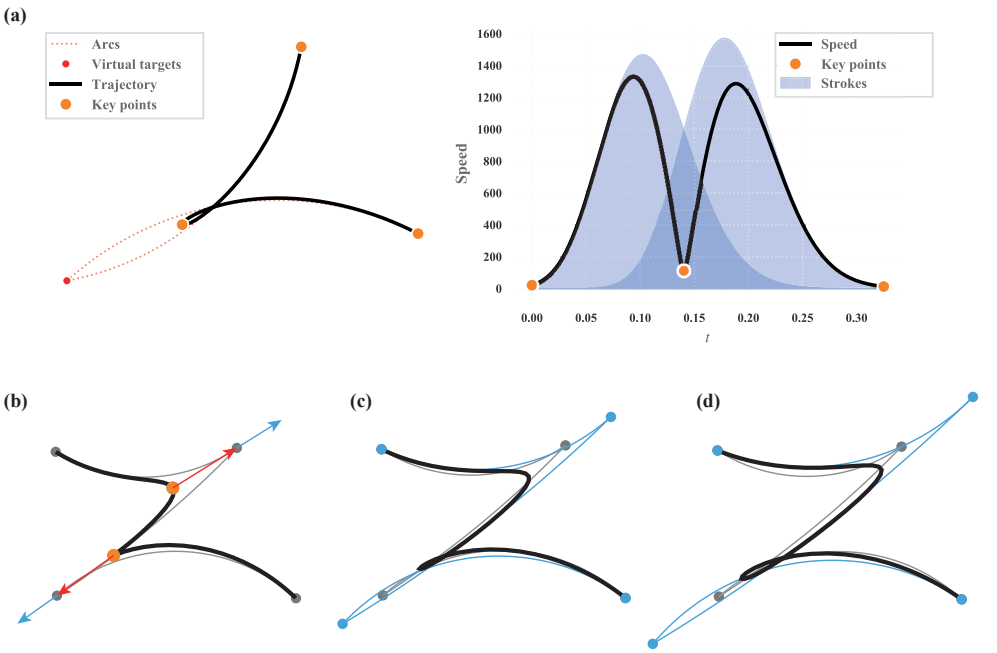


Figure 4.8: Key-point adjustment. **(a)** Key-points (orange circles) overlaid on the trace (left) and speed profile (right) of a trajectory generated with two lognormal primitives. **(b)** The adjustment vectors (red arrows) go from the key-points to the virtual target. These are scaled by $\lambda_p = 0.7$ (blue arrows) and applied to the virtual targets. **(c)** Result of one adjustment step. **(d)** Result of a second adjustment steps, after re-computing the key-points from the previous configuration.

with a few iterations of a Newton scheme.

The adjusted trajectory is computed with a new sequence of virtual target positions $\hat{\boldsymbol{p}}_i$ such that the locations of the corresponding key-points $\boldsymbol{x}(\tau_i)$ in the resulting trajectory approach the originally specified virtual targets $\boldsymbol{p}_i$ (Figure 4.8b). We first let $\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i$ and then, for a user specified number of iterations, we adjust the new virtual target positions with:

$$\hat{\boldsymbol{p}}_i \leftarrow \hat{\boldsymbol{p}}_i + \lambda_p(\boldsymbol{p}_i - \boldsymbol{x}(\tau_i)) \,. \tag{4.17}$$

In practice, a small number of iterations prove sufficient and the procedure runs in real-time (Figure 4.8c, d) since each step only requires computing the trajectory at $M-1$ key-point locations. This allows a user to interactively determine the desired amount of adjustment by setting the number of iterations and the parameter λ_p. As an example of this procedure, Figure 4.9 demonstrates how a few adjustment steps can be used to address the issue with parameter scaling that was observed in Figure 4.7d. This same procedure is also useful when

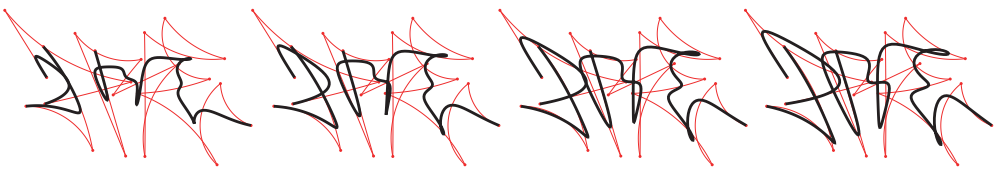


Figure 4.9: Key-point adjustment for Δt_i parameters scaled by a factor of 0.4. From left to right, the unadjusted trajectories followed by three key-point adjustment steps with $\lambda_p = 0.7$.

combined with a UI, since it forces the curvature extrema of the trajectory to be closer to the virtual targets, resulting in a behavior that is closer to an interpolation.

4.4.2.2 Parameter exaggeration

More sophisticated stylisation results can be achieved by exploiting the $\Sigma\Lambda$ parameter structure to perform an "exaggeration" of the kinematic features with a procedure inspired by the work of Brennan (1985), who extrapolates facial features from an average to generate caricatures. We accentuate deviations of the time-overlap Δt_i and curvature δ parameters from their respective mean values $\bar{\Delta t}, \bar{\delta}$ with

$$\Delta t_i \leftarrow \Delta t_i + k_{\Delta t}\left(\Delta t_i - \bar{\Delta t}\right) \tag{4.18}$$

$$\delta_i \leftarrow \delta_i + k_{\delta}\left(\delta_i - \bar{\delta}\right) \tag{4.19}$$

where positive values of the parameters $k_{\Delta t}, k_{\delta}$ exaggerate differences of $\Delta t_i, \delta_i$ from their respective means, while negative values decrease the differences, and null values leave the parameters unaffected (Figure 4.10).

4.5 Stroke generation and animation

One of the advantages of generating curves through the simulation of a movement is that the smooth kinematics can be exploited to drive the implementation of expressive rendering methods. For example, in prior work the same type of handwriting model has been exploited to generate realistic renditions of signature pen traces (Ferrer et al., 2015) using an ink deposition model (Franke and Rose, 2004). The brush model that follows, is aimed at achieving various effects that are evocative of instances of ink-calligraphy and graffiti made with markers or spray paint by exploiting the kinematics captured by the $\Sigma\Lambda$ model. The method builds upon the assumption that the amount of paint deposited is inversely proportional to the speed of the drawing tool (Figure 4.11). This model is simple while providing visually appealing patterns which are approximately similar to the ones produced with physically more accurate but also more complex models of a brush or pen.

In order to generate a variably smooth brush texture, we use once more the error func-

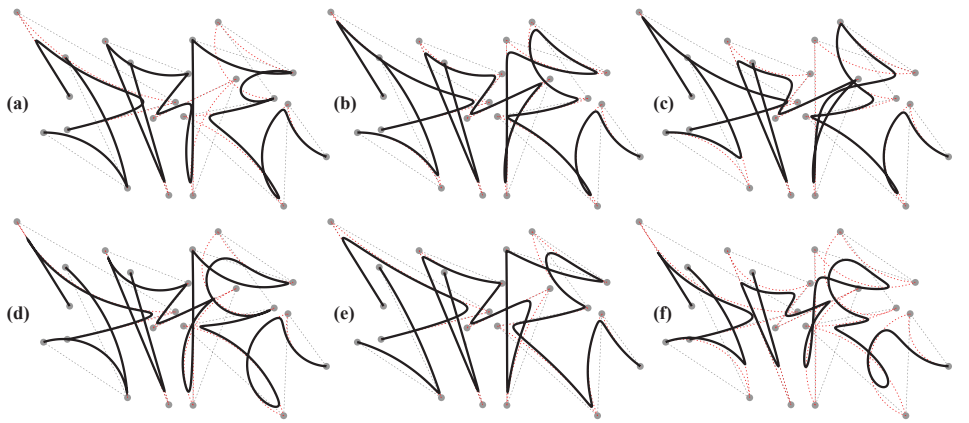


Figure 4.10: Parameter exaggeration. **(a)** no exaggeration. **(b)** $k_\delta = 0.0, k_{\Delta t} = 2$. **(c)** $k_\delta = 0.0, k_{\Delta t} = 2$. **(d)** $k_\delta = 0.5, k_{\Delta t} = 0$. **(e)** $k_\delta = -0.5, k_{\Delta t} = 0$. **(f)** Combination of Δt_i scaled by a factor of 0.5 and exaggeration with $k_\delta = 0.5, k_{\Delta t} = 0.3$.

Figure 4.11: Kinematics-based brush rendering of $\Sigma\Lambda$ trajectories: the corresponding motor plan (in red on the left) and renderings with different kinematic dependent brushes.

tion to obtain a "hat" curve (Figure 4.12a) with the following equation:

$$\phi_b(x) = \frac{1}{2} + \frac{1}{2}\text{erf}[\alpha_b(1-x)] \quad , \tag{4.20}$$

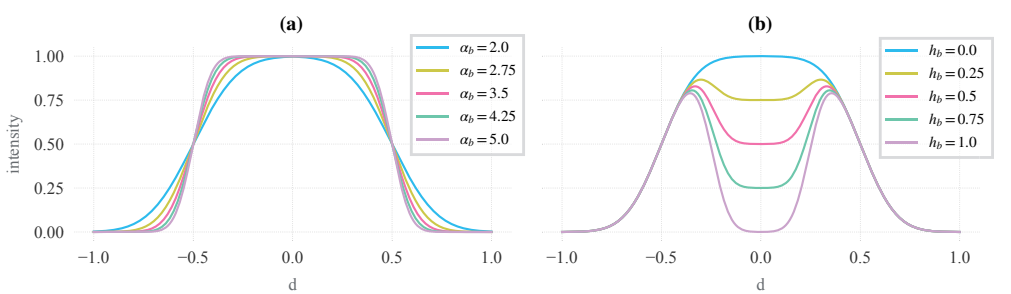


Figure 4.12: "Hat" functions for brush generation with different parameters. **(a)** single curve. **(b)** combination of two curves resulting in a decrease of intensity near the brush center.

where the parameter α_b determines the top hat flatness of the curve.

To mimic the effects of certain spray nozzles or marker nibs that diffuse less paint near center we compute the curve with $\phi_b(x) * (1 - o_b\phi_b(2x))$ with o_b a user configurable parameter that is inversely proportional to the brush intensity near its center (Figure 4.12b). A variably sized and rotated 2D brush texture is produced by using the hat curve with normalised coordinates $(u, v) \in [0,1]$ to obtain the distance to a superellipse with

$$\sqrt{\left|\frac{u\cos\theta_b - v\sin\theta_b}{w_b}\right|^{\beta_b} + \left|\frac{u\sin\theta_b + v\cos\theta_b}{h_b}\right|^{\beta_b}} \quad , \tag{4.21}$$

where θ_b determines the brush rotation and w_b, h_b, β_b respectively determine the relative width and height of the brush and the shape of the superellipse. Figure 4.13 shows different examples of brush textures with the corresponding parameters.

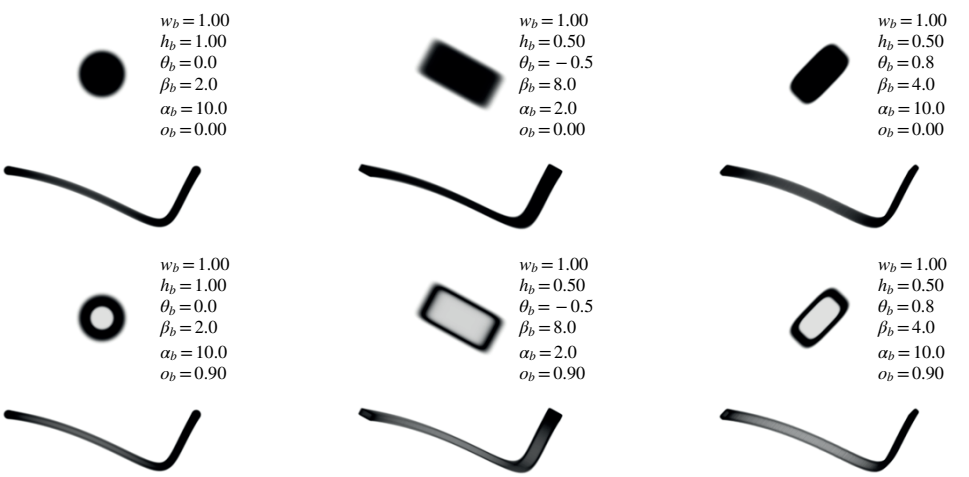


Figure 4.13: Different brush textures, with the corresponding parameters and an example trajectory

We further use the traditional "dabbing" a.k.a. "stamping" procedure to sweep the brush along the trajectory and scale its size as an inverse function of speed:

$$r(t) = r_{\min} + (r_{\max} - r_{\min})\exp\left(-\frac{\bar{v} + \left|\dot{\boldsymbol{p}}(t)\right|}{\bar{v}}\right) \tag{4.22}$$

i.e. scaled by the mean of the speed for the whole trajectory, $\bar{v}$ (Figure 4.14). The brush size is varied within a range $[r_{min}, r_{max}]$, which allows to adjust the amount of speed dependent scaling in the generated image. The speed $\left|\dot{\boldsymbol{p}}(t)\right|$ can be exactly computed by using the original form of the $\Sigma\Lambda$ model (Plamondon et al., 2014) or approximated by computing by forward

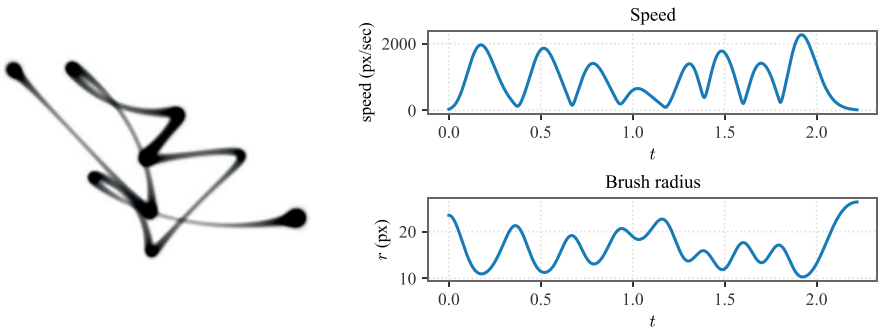


Figure 4.14: Dabbing a brush as a function of speed (top right) with a variable width (bottom right).

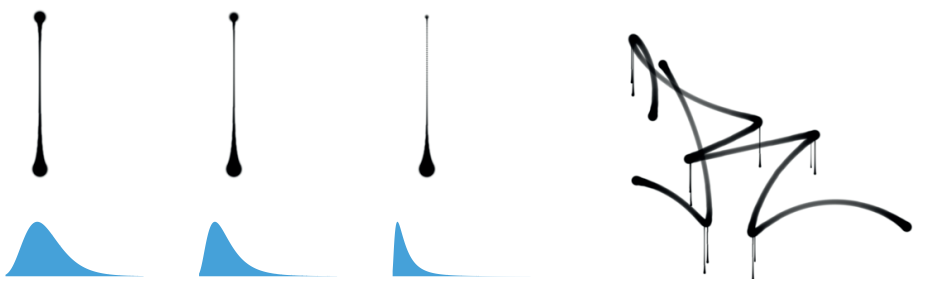


Figure 4.15: Lognormal drips. Left to right: Three variations of a drip generated with thickness determined by a lognormal (blue profile) with respective shape parameters $A_c = (0.1, 0.25, 0.5)$, followed by a swept stroke rendering of a letter "R" with drips.

differencing the trajectory $\boldsymbol{x}(t)$ which proves faster and sufficiently accurate for this application.

Drips With the similar assumption that slower movements produce a higher deposition of ink/paint, we can add a random drip effect where the speed is under a user defined threshold (Figure 4.15). This mimics a feature that can often be seen in instances of tags made with an ink marker or spray paint. We use a purely visual trick to mimic the appearance of a drip running down a surface, where we determine the thickness of the drip using equation (4.22) with the speed profile provided by a single lognormal.

Animation and Fabrication The physiologically plausible kinematics produced by the $\Sigma\Lambda$ model can also be exploited to easily produce natural looking stroke animations of a trajectory. This is achieved by incrementally sweeping a brush texture along a uniform-time sampling of the trajectory, since the distance between samples reflects the smooth kinematics generated by the model. The same technique can be exploited to produce smooth motion paths for virtual (human) characters, or for fabrication, or robotic devices (Figure 4.16) (Berio et al., 2016).

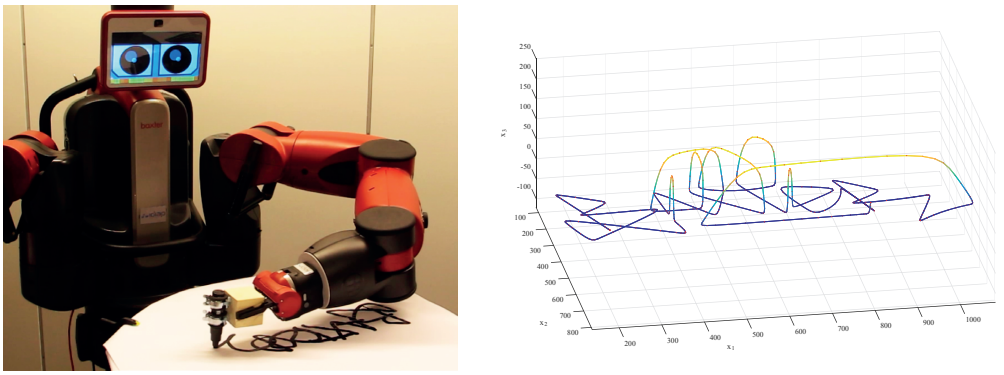


Figure 4.16: *Left*: A compliant robot reproducing a tag created with the ΣΛ model. *Right*: Corresponding trajectory with the 3D motion of the marker, computed with the ΣΛ model and with the addition of a third coordinate permitting smooth transitions (pen-up). This results in helical rather than circular-arc sub-movements. Refer to Berio et al. (2016) for technical details.

4.6 Conclusion

We have shown how the ΣΛ model, a physiologically plausible model of handwriting movements, can be adapted to the tasks of interactive curve generation, stylisation and rendering . The parameters of the ΣΛ model have a clear physiological interpretation in terms of both planning and trajectory formation. At the planning level, the virtual targets correspond to a motor plan that guides the superposition of a series of ballistic sub-movements. Each submovement is characterised by a set of kinematic parameters that determine the fine evolution of a kinematic realisation of the motor plan. For the use case of calligraphic stylisation, the kinematic parameters are $\Theta = \{\Delta t_i, \delta_i\}$ for the $\omega\Sigma\Lambda$ parameterisation and $\Theta = \{\Delta t_i, s_{0i}, s_{1i}\}$ for the $\omega\mathscr{E}\Sigma\Lambda$ parameterisation, while the parameters Ac_i and T_i are kept constant to user-defined values.

In the pattern recognition and handwriting analysis domains the ΣΛ representation has proved to be useful to accurately reconstruct and characterise handwriting movements (Plamondon et al., 2014; Ferrer et al., 2018), with applications ranging from forgery detection (Gomez-Barrero et al., 2015) to the study of motor-related problems such as Parkinson's disease(Plamondon et al., 2013). In our application, the ΣΛ parameterisation becomes especially useful to interactively edit trajectories with an interface that is similar to conventional CGAD methods, and to generate variations of a trajectory that are similar to the ones that would be produced by a human.

The parameterisation of the model also comes with a drawback, when it comes to automatically generating calligraphic stylisations of a motor plan. With the previously described approach, a user is required to explicitly set the kinematic parameters through an interac-

tive user interface. However, choosing these parameters automatically remains challenging and the $\Sigma\Lambda$ model per-se does not provide a systematic way to do so. In the next chapter, we address this limitation with a different parameterisation and a different trajectory formation method that is based on optimisation. In Chapters 8 and 9 we will go back to the $\Sigma\Lambda$ model and exploit its structure to determine the kinematic parameters automatically with an example-driven procedure. Combining optimisation methods with the $\Sigma\Lambda$ remains a promising area of future work, and we will expand further on this topic in the conclusion of Chapter 8 and in Chapter 12.

Chapter 5

Calligraphic stylisation: Minimal intervention control

This chapter is largely based on published work developed in a collaboration between myself, Prof. Frederic Fol Leymarie and Dr. Sylvain Calinon (affiliation: Idiap, Switzerland). The collaboration resulted in two conference papers (Berio et al., 2017b,c) and one book chapter (Berio et al., 2020a). This body of work was initially based on optimal control techniques developed by Dr. Calinon for programming by demonstration applications in robotics, which include the stochastic solution to the discrete optimal control problem discussed in Section 5.1.5 and the solution for multiple tracking references discussed in Section 5.1.7 Calinon (2016b). My contribution includes the development and implementation of the remaining methods and formulations discussed in this chapter, which I developed with the specific aim of interactive curve editing and calligraphic curve generation. This chapter includes additional details and extensions that I have developed in the context of this thesis and are not included in the our previously published works. This includes an updated formulation that enables arbitrary sampling quality of trajectories Section 5.1.1, and an improved derivation of periodic trajectories Section 5.1.6.

In the previous chapter, we have seen how the $\Sigma\Lambda$ model can be used as an interactive curve generation tool, and how its physiologically plausible parameterisation can be exploited to generate variations and stylisations of a trajectory that mimic the variability that would be seen in multiple instances of human drawing or writing. In this chapter, we adopt a complementary approach, in which a trajectory is explicitly defined in terms of its desired precision and variability. The result is a versatile trajectory generation method that does not generate one, but a family of trajectories that can be stochastically sampled from a probability distribution. As we shall see, the same stochastic formulation also allows for an intuitive

interface, which can be used to determine the fine curvilinear evolution of a trajectory, and finally to generate traces that are qualitatively similar to instances of graffiti tags and calligraphy.

The input to the method is again a motor plan, but this time it is augmented with a *mixture of Gaussians* (MoG) describing a spatial distribution. The output is a distribution of smooth trajectories, with kinematics that are similar to the ones that typically characterize human hand motions and variations determined by the input distribution. We generate a trajectory by forcing a dynamical system to track the centers of Gaussians with a precision determined by their respective covariances. The trajectory evolution is determined by optimization, with an objective formulated as a trade-off between tracking accuracy and control effort. Control effort is expressed as the square magnitude of position derivatives, preferably of high order, e.g. jerk (3rd) or snap (4th), which results in smooth trajectories that are consistent with the minimum-square-derivative hypothesis (Flash and Hogan, 1985), but also obey a minimal intervention principle (Todorov and Jordan, 2002b), where deviations from maximal smoothness are corrected only when the required precision is high. As a result, we will refer to the trajectory generation method as *minimal intervention control* (MIC).

5.1 Trajectory Generation

The input to our method is a sequence of multivariate Gaussians $\mathcal{N}\left(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)$ defined in a Cartesian space of dimension D, which is equivalent to a mixture of Gaussians (MoG) with uniform weights. The output of the method is a distribution $\mathcal{N}\left(\boldsymbol{y}, \boldsymbol{\Sigma}_{\boldsymbol{y}}\right)$ of smooth motions that track the centers $\boldsymbol{\mu}_i$ with a precision defined by the corresponding covariances $\boldsymbol{\Sigma}_i$. The centers $\boldsymbol{\mu}_i$ coincide with the vertices $\boldsymbol{p}_i$ of a motor plan. At the same time, the covariance structure of the MoG provides explicit control over the variability and smoothness of the trajectory in the region of each vertex, together with local or global control of the curvilinear evolution of the trajectory.

Trajectories are generated by optimizing the evolution of a dynamical system that tracks each MoG component sequentially for a given amount of time. A decrease in the variance of

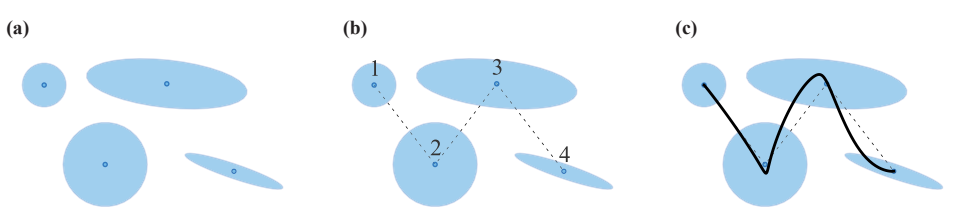


Figure 5.1: The trajectory generation method in a nutshell. **(a)** An input GMM is considered as **(b)** a sequence. **(c)** These ordered components are then used to guide the evolution of a dynamical system.

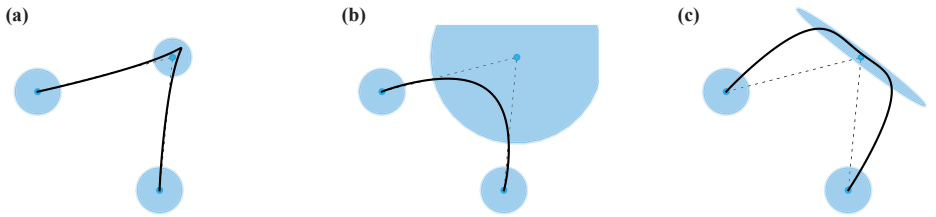


Figure 5.2: Variations of a trajectory by manipulating one covariance matrix. **(a)** using an isotropic covariance with low variance (high precision). **(b)**, an increase in variance produces a smoothing effect. **(c)** a full (anisotropic) covariance can be used to force the trajectory to remain in a flat region of space.

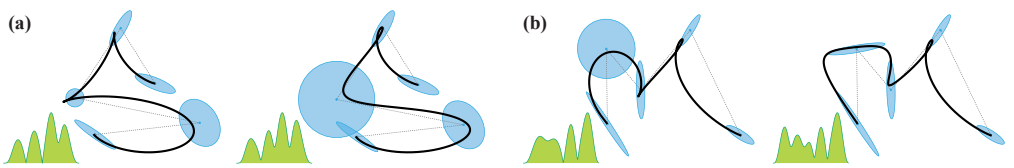


Figure 5.3: **(a)**, smoothing effect of increasing the variance of a Gaussian. **(b)**, manipulating the trajectory evolution with full covariances. Below each trajectory, its corresponding speed profile.

a component corresponds to an increased precision requirement, and thus forces the trajectory to pass nearer the component center (Figure 5.2a). A sufficiently low variance then produces an interpolatory behavior. An increase in the variance corresponds with a lower precision requirement, and thus produces a smoothing effect that is similar to the one achieved with smoothing splines (Figure 5.2b). However, the use of full covariances allows more complex spatial constraints to be captured, such as forcing a movement to follow a given direction or to pass through a narrow region of space (Figure 5.2c). The resulting trajectories are smooth and have kinematics that are similar to the ones that would be seen in a movement made by a drawing hand, with desirable features such as bell shaped speed profiles and an inverse relation between speed and curvature (Figure 5.3).

5.1.1 Dynamical system

We model the spatial evolution of trajectory with a linear time invariant (LTI) system of order n. The evolution of each coordinate x along a trajectory is governed by the state equations

$$\dot{\boldsymbol{x}} = \bar{\boldsymbol{A}}\boldsymbol{x} + \bar{\boldsymbol{B}}u,$$
$$\boldsymbol{y} = \bar{\boldsymbol{C}}\boldsymbol{x}$$

where the state

$$\boldsymbol{x} = \left[x, \dot{x}, \ddot{x} \ldots, {}^{(n-1)}x\right]^{\top} \in \mathbb{R}^n$$

concatenates position and its derivatives up to order $n-1$. The system matrices $\bar{\boldsymbol{A}}$, $\bar{\boldsymbol{B}}$ and $\bar{\boldsymbol{C}}$ describe the time invariant response of the system to an input command u, and are given by the canonical form

$$\bar{\boldsymbol{A}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad \bar{\boldsymbol{B}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^{n \times 1}, \quad \bar{\boldsymbol{C}} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}^{\top} \in \mathbb{R}^{1 \times n} \tag{5.1}$$

where $\bar{\boldsymbol{C}}$ is a "sensor matrix" that determines what elements of the state are observed in a feedback system. This system definition consists of a chain of n integrators commanded by its n-th order derivatives. As an example, a linear system of this type with order 2 is is equivalent to a spring-mass-damper system that is controlled with acceleration commands.

5.1.1.1 Discretisation

To compute a trajectory that tracks the centers of M Gaussians, we use a discretisation of the system with N time steps of constant duration Δt. This corresponds to a control signal consisting of $N-1$ piecewise constant commands $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N \in \mathbb{R}^D$ and results in a sequence of N points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$ along the trajectory. We use an *exact discretisation* (Haugen, 2005) of the system, and this allows to retrieve trajectories of arbitrary precision once an initial, and possibly sparse, estimate has been computed. In the general case this is given by:

$$\boldsymbol{A}_d = \boldsymbol{e}^{\bar{\boldsymbol{A}}\Delta t} \quad \text{and} \quad \boldsymbol{B}_d = \int_0^{\Delta t} \boldsymbol{e}^{\bar{\boldsymbol{A}}t} \, dt \, \bar{\boldsymbol{B}} \quad .$$

For the specific case of a chain of integrators the integral can easily be computed in closed form with (DeCarlo, 1989, pg. 215):

$$\boldsymbol{e}^{\begin{bmatrix} \bar{\boldsymbol{A}} & \bar{\boldsymbol{B}} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{\top}} = \begin{bmatrix} \boldsymbol{A}_d & \boldsymbol{B}_d \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \quad .$$

which corresponds to a *zero order hold* (ZOH) discretisation of $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$.

We determine the points at each time step t of a trajectory with the discrete state equa-

tions:

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t \tag{5.2}$$

$$\boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{x}_t \tag{5.3}$$

where the state $\boldsymbol{x}_t$ concatenates the D coordinates of a position with its derivatives up to the order $n-1$ with:

$$\boldsymbol{x}_t = \left[x_1, \ldots, x_D, \dot{x}_1, \ldots, \dot{x}_D \ldots, \overset{(n-1)}{x_1}, \ldots, \overset{(n-1)}{x_D}\right]^\top \in \mathbb{R}^{Dn} \quad . \tag{5.4}$$

The discrete time system matrices $\boldsymbol{A} \in \mathbb{R}^{Dn \times Dn}$, $\boldsymbol{B} \in \mathbb{R}^{Dn \times D}$ and $\boldsymbol{C} \in \mathbb{R}^{D \times Dn}$ are block matrices with one block for each entry of $\boldsymbol{A}_d, \boldsymbol{B}_d, \bar{\boldsymbol{C}}$. Each block is given by a $D \times D$ identity matrix $\boldsymbol{I}$ multiplied by the scalar in the corresponding entry. This can be conveniently computed with the Kronecker product operator $\otimes$ with:[1]

$$\boldsymbol{A} = \boldsymbol{A}_d \otimes \boldsymbol{I}, \quad \boldsymbol{B} = \boldsymbol{B}_d \otimes \boldsymbol{I} \quad \text{and} \quad \boldsymbol{C} = \bar{\boldsymbol{C}} \otimes \boldsymbol{I} \quad .$$

It can be shown that the system formulation in equation (5.1), together with a piecewise constant control sequence, results in a polynomial spline of degree at least n. The spline is C^{n-1} continuous and has one knot for each step in the command sequence (Kano et al., 2005). As a result, given a possibly sparse sequence of N commands $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ and an initial state $\boldsymbol{x}_1$, we can efficiently retrieve a smooth trajectory $\boldsymbol{x}(t)$ with arbitrary precision. Since the discretisation in equation (5.4) is exact, this can be done by computing the discretised system matrices with a reduced time step $\Delta t'$ and iteratively computing the trajectory with equation (5.2), starting from $\boldsymbol{x}_1$ and with each command $\boldsymbol{u}_t$ being kept constant for $\frac{\Delta t}{\Delta t'}$ time steps.

5.1.2 Optimization objective

We generate N samples along a trajectory by computing an optimal controller that minimizes a quadratic cost, which penalizes a trade-off between deviations from a reference state sequence $\{\bar{\boldsymbol{x}}_t\}_{t=1}^N$ (*tracking cost*) and the magnitude of a control command sequence $\{\boldsymbol{u}_t\}_{t=1}^{N-1}$ (*control cost*). The optimization objective is expressed with the summative cost:

$$J = \sum_{t=1}^{N} (\bar{\boldsymbol{x}}_t - \boldsymbol{x}_t)^\top \boldsymbol{Q}_t (\bar{\boldsymbol{x}}_t - \boldsymbol{x}_t) + \sum_{t=1}^{N-1} \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t \quad , \tag{5.5}$$

subject to the constraint of the linear system defined in equation (5.2) and where $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ are positive semi-definite weight matrices that determine the tracking and control penalties for each time step. The linear constraint guarantees that the output of the method is a trajec-

[1] The *kron* command is available in most linear algebra packages such as Matlab or NumPy.

tory that has continuous derivatives up to the order $n-1$. As the discretisation time step Δt tends to zero, the resulting trajectory becomes an increasingly accurate approximation of a spline of degree $(2n-1)$ with $(2n-2)$ continuous derivatives (Zhang et al., 1997).

The combination of a linear system with this type of optimization objective is commonly used in process control and robotics applications, where it is known as discrete Linear Quadratic Tracking (LQT) and corresponds to the linear case of Model Predictive Control (MPC) (Zeestraten et al., 2016b). This results in a standard optimization problem that can be solved iteratively or in batch form and produces an optimal controller or control command sequence. In typical control settings, the optimization is performed iteratively over a time horizon of observations, and is thus commonly known as receding horizon control. However, for the intended use case of curve design, we can apply the optimization to the full duration of the desired trajectory. With the appropriate formulation of the reference, this results in a flexible curve generation method that can be used similarly to more conventional curve generation methods.

5.1.3 Tracking formulation

We formulate the reference state and weights for the optimization objective, by considering a decomposition of a movement that tracks M Gaussians, into $M-1$ ballistic sub-movements. With an assumption of *local isochrony*, we assign each sub-movement a fixed number of time steps T_s resulting in a sub-movement duration of $T_s \Delta t$. This gives a tracking reference with $N = (M-1)T_s + 1$ time steps and assigns each Gaussian a *passage time* τ_t, with $\tau_{i+1} - \tau_i = T_s$, and with $\tau_1 = 1$ and $\tau_M = N$. Each Gaussian is also assigned an activation function:

$$h_i(t) = \frac{\phi_i(t)}{\sum_{j=1}^{M} \phi_j(t) + e^{-6}} \quad \text{with} \quad \phi_i(t) = \exp\left(-\frac{(t-\tau_i)^2}{2\hat{\sigma}_h^2}\right) \quad \text{and} \quad \hat{\sigma}_h = \frac{3T_s}{2}\sigma_h \quad , \tag{5.6}$$

computed in terms of a radial basis function (RBF) $\phi_i(t)$, where $\sigma_h \in (0,1]$ is a global parameter which defines the *time interval* covered by each state, and the denominator term e^{-6} avoids divisions by zero while guaranteeing that $\sigma_h = 1$ results in the time intervals covering all time steps.

The reference states and weights are then generated by assigning to each time step the state for which $h_i(t) > 0.5$ (Figure 5.4, second row) with:

$$\tilde{\boldsymbol{x}}_t = \boldsymbol{C}^\top \boldsymbol{\mu}_i \quad \text{and} \quad \boldsymbol{Q}_t = \boldsymbol{C}^\top \boldsymbol{\Sigma}_i^{-1} \boldsymbol{C} \quad . \tag{5.7}$$

With this formulation the derivatives of the trajectory are fully determined by the optimization procedure, which is expressed by setting the corresponding precision terms $\boldsymbol{Q}_t$ to zero. Intuitively, a zero entry in $\boldsymbol{Q}_t$ means that the optimization has no precision requirements for the corresponding state entry and thus is free to enforce the smoothness require-

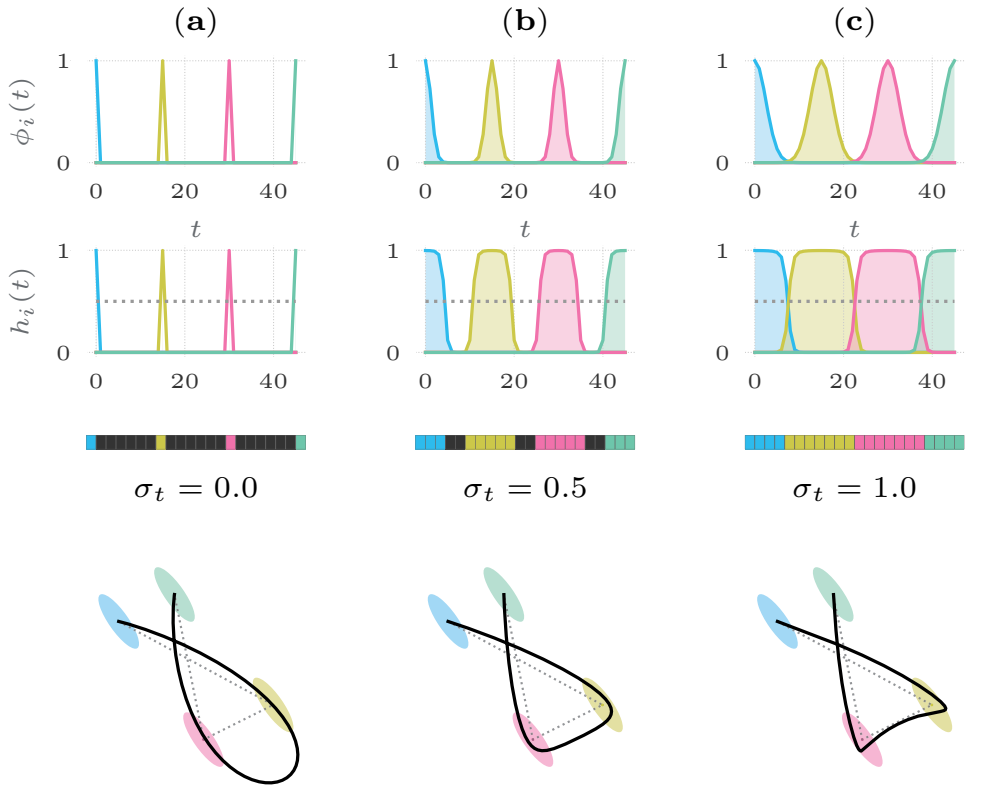


Figure 5.4: Effect of different activation sequences with the same set of Gaussians (color coded). The first and second row show respectively the RBF term ϕt_i for each Gaussian and the resulting activation $h_i(t)$. The third row shows in color the time steps covered by each state. The fourth row shows the resulting trajectories and their Gaussians.

ment expressed in the second term of the cost function. In typical applications, the tracking weights $\boldsymbol{Q}_t$ are defined as diagonal matrices, such as the case of smoothing splines. This corresponds to a penalty in terms of the Euclidean distance to a reference state. In our stochastic formulation, the weights are expressed as full precision matrices, which corresponds to a penalty in terms of the Mahalanobis distance[2] to the reference state. When it is desirable to force the movement to a full stop, this can be done by setting $\boldsymbol{Q}_N = \boldsymbol{I}$ and all the derivative terms in $\bar{\boldsymbol{x}}_N$ to zero.

Increasing the value of σ_h increases the time interval covered by a state, with $\sigma_h = 1$

[2]Corresponding to the Euclidean distance in a transformed coordinate system determined by the mean and covariance of a Gaussian (Murphy, 2012).

resulting in a step-wise reference that fully covers the time steps of the trajectory (Figure 5.4c). This increases the influence of the MoG covariances on the resulting trajectory, and allows a user to specify curvilinear trends and variability for longer segments of the trajectory. As the parameter σ_h tends to zero, $\phi_i(t)$ will converge to a Delta function (Figure 5.4a), which will result in $\boldsymbol{Q}_t$ being non-zero only in correspondence with each passage time τ_i. This gives smoother trajectories that interpolate the control-points. In general, a smaller time interval will result in a sparser tracking cost in the objective. This increases the influence of the control cost and potentially facilitates the addition of objectives and constraints to the optimization.

5.1.4 Control weights

The weight matrices $\boldsymbol{R}_t$ define a penalty on the amplitude of control commands. Typically this cost is formulated as a constant diagonal term that is inversely proportionally to the maximum square norm of the control command, where a larger term produces smoother trajectories (Figure 5.5.a). However, manually setting this value can be counter-intuitive, especially when varying also the order of the system (Figure 5.5.b). In order to achieve approximately equal tracking performance across different system orders (Figure 5.5.c), we express the control cost in terms of a *maximum allowed displacement* $\Delta_{\max}$. To compute the corresponding terms in $\boldsymbol{R}_t$ we exploit the frequency gain of the system, which for the case of the chain of integrators simply reduces to $1/\omega^n$ (refer to Appendix C.1 for additional details and derivations), where ω is the angular frequency of a sinusoidal input. The diagonal weights in $\boldsymbol{R}_t$ are given by

$$\boldsymbol{R}_t = \frac{1}{(\omega^n \Delta_{\max})^2} \boldsymbol{I} \qquad \text{and} \qquad \omega = \frac{2\pi}{T_s \Delta t} \quad , \tag{5.8}$$

where the frequency ω is empirically set using the duration of one sub-movement as a period. Lower values of $\Delta_{\max}$ tend to smooth the trajectory, while higher values generate sharper paths. Because the cost function is defined as a trade-off between tracking and control cost, it is in practice possible to achieve the same effect by either increasing the variance of the Gaussians or decreasing the value of $\Delta_{\max}$.

5.1.5 Stochastic solution

The optimal trajectory can be retrieved iteratively using dynamic programming (Calinon, 2016a), or in batch form by solving a large ridge regression problem. While we leave details on the former iterative solution to Appendix C.2, here we focus on the latter, which results in a more compact solution and provides great additional flexibility, such as a straightforward probabilistic interpretation of the result and the possibility to generate periodic trajectories.

To compute the solution, we exploit the time invariance of the system, and express all

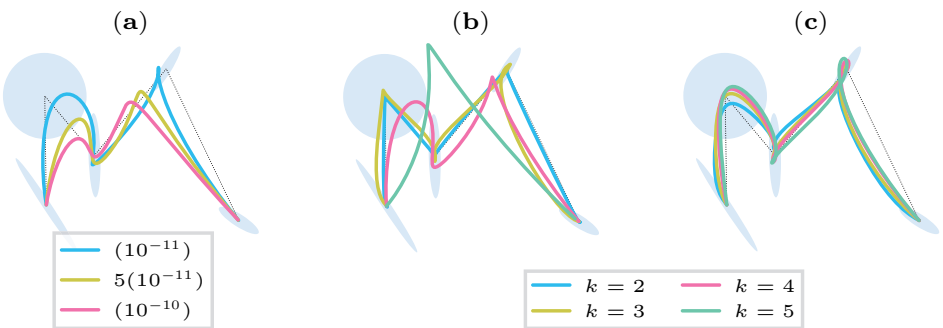


Figure 5.5: Order independent control weight. **(a)** Trajectories for a 4th order system, for different diagonal values of r with $\boldsymbol{R}_i = r\boldsymbol{I}$. **(b)** Varying the system order while keeping r constant. **(c)** Varying the system order with a displacement based weighting, using $\Delta_{\max} = 25$.

future states as a function of the initial state $\bar{\boldsymbol{x}}_1$ with:

$$\hat{\boldsymbol{x}} = \boldsymbol{S}_x \bar{\boldsymbol{x}}_1 + \boldsymbol{S}_u \boldsymbol{u} \quad , \tag{5.9}$$

where

$$\boldsymbol{S}_x = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix} \quad \text{and} \quad \boldsymbol{S}_u = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{AB} & \boldsymbol{B} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \dots & \boldsymbol{B} \end{bmatrix} \quad .$$

We then express the objective (5.5) in matrix form as:

$$J = (\bar{\boldsymbol{x}} - \hat{\boldsymbol{x}})^\top \boldsymbol{Q} (\bar{\boldsymbol{x}} - \hat{\boldsymbol{x}}) + \boldsymbol{u}^\top \boldsymbol{R} \boldsymbol{u} \quad , \tag{5.10}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are large block matrices with $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ along their block diagonals, while $\bar{\boldsymbol{x}}$, $\hat{\boldsymbol{x}}$ and $\boldsymbol{u}$ are column vectors respectively stacking the reference, state and control commands for each time step. Substituting (5.9) into (5.10), differentiating with respect to $\boldsymbol{u}$ and setting to zero results in a least squares solution gives:

$$\boldsymbol{u} = \underbrace{\left(\boldsymbol{S}_u^\top \boldsymbol{Q} \boldsymbol{S}_u + \boldsymbol{R}\right)^{-1}}_{\boldsymbol{\Sigma}_u} \boldsymbol{S}_u^\top \boldsymbol{Q} (\bar{\boldsymbol{x}} - \boldsymbol{S}_x \bar{\boldsymbol{x}}_1) \quad , \tag{5.11}$$

which is then substituted back into (5.9) to generate a trajectory.

5.1.5.1 Stochastic sampling

We can see that equation (5.11) is equivalent to a ridge regression solution where $\boldsymbol{R}$ effectively acts as a Tikhonov regularization term, giving a global smoothing effect on the generated trajectory. From a probabilistic perspective, $\boldsymbol{R}$ corresponds to a Gaussian prior on the deviations of the control commands from $\mathbf{0}$. The minimization of equation (5.10) can then be interpreted as the product of two Gaussians:

$$\mathcal{N}(\boldsymbol{u}, \boldsymbol{\Sigma_u}) \propto \mathcal{N}\left(\boldsymbol{S_u}^{-1}(\bar{\boldsymbol{x}} - \boldsymbol{S_x}\bar{\boldsymbol{x}}_1), \boldsymbol{S_u}^{\top}\boldsymbol{Q}\boldsymbol{S_u}\right)\mathcal{N}(\mathbf{0}, \boldsymbol{R}) \quad , \tag{5.12}$$

describing a distribution of control commands with center $\boldsymbol{u}$ and covariance $\boldsymbol{\Sigma_u}$. By using the linear relation (5.9) the distribution in control space can also be interpreted as a *trajectory distribution*:

$$\mathcal{N}\left(\boldsymbol{y}, \boldsymbol{\Sigma_y}\right) \quad \text{with} \quad \boldsymbol{\Sigma_y} = \boldsymbol{S_u}\boldsymbol{\Sigma_u}\boldsymbol{S_u}^{\top} \quad .$$

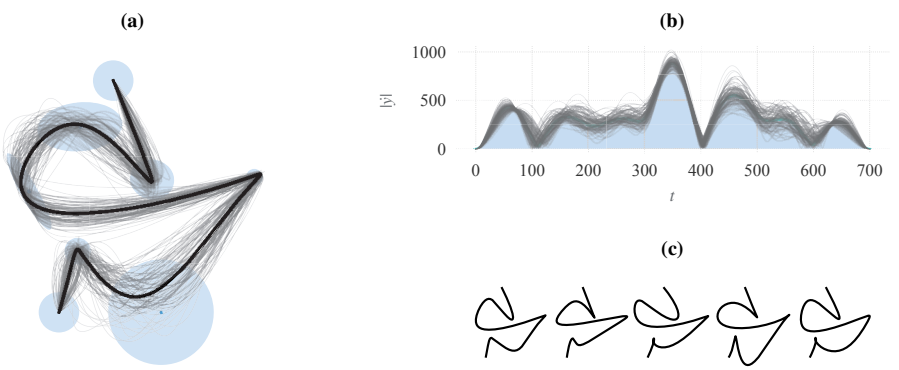


Figure 5.6: Stochastic sampling from the trajectory distribution. **(a)** GMM with corresponding trajectory overlaid with samples from the trajectory distribution. **(b)** Sampled speed profiles. **(c)** Single samples from the trajectory distribution.

This formulation results in a generative model of trajectories, which can be used to generate variations that are similar to the ones that would be seen in multiple instances of human writing or drawing. Because of its lower dimensionality, it is preferable to generate variations at the control level, which can be done by exploiting the eigen decomposition:

$$\boldsymbol{\Sigma_u} = \boldsymbol{V_u}\boldsymbol{D_u}\boldsymbol{V_u}^{\top} \quad , \tag{5.13}$$

with $\boldsymbol{V_u}$ a matrix with all eigenvectors along the columns and $\boldsymbol{D_u}$ a matrix with the corresponding eigenvalues along the diagonal. We can then generate samples around the average

commands sequence $\boldsymbol{u}$ with:

$$\boldsymbol{u}' \sim \boldsymbol{u} + \boldsymbol{V}_{\boldsymbol{u}} \boldsymbol{D}_{\boldsymbol{u}}^{\frac{1}{2}} \mathcal{N}(\boldsymbol{0}, \sigma_u \boldsymbol{I}) \quad ,$$

with σ_u a user-defined parameter representing the variation in the samples. The resulting trajectories can then easily be retrieved by plugging the samples $\boldsymbol{u}'$ into equation (5.9).

5.1.6 Periodic motions

In order to generate periodic motions, we require a reference state sequence $\bar{\boldsymbol{x}}$. This can be obtained by considering a sequence of $M+1$ Gaussians, with the first one repeated at the end of the sequence. We then reformulate the LQT objective so it optimises also for the initial state $\boldsymbol{x}_1$ and by adding an equality constraint on the initial and final state of the trajectory. This constraint can be formulated with the linear relation:

$$\boldsymbol{K}\boldsymbol{x} = \boldsymbol{K}(\boldsymbol{S}_{\boldsymbol{x}}\bar{\boldsymbol{x}}_1 + \boldsymbol{S}_{\boldsymbol{u}}\boldsymbol{u}) = \boldsymbol{0} \quad , \tag{5.14}$$

with $\boldsymbol{K}$ a matrix in $\mathbb{R}^{Dn \times DnN}$ with zero blocks everywhere except for the first and the last time-steps, which are set to $-\boldsymbol{I}$ and $\boldsymbol{I}$ respectively. Adding the constraint to equation (5.10) results in the Lagrangian:

$$\mathscr{L}(\boldsymbol{u}, \boldsymbol{x}_1, \boldsymbol{\lambda}) = J + \boldsymbol{\lambda}^\top \boldsymbol{K}\hat{\boldsymbol{x}} \quad . \tag{5.15}$$

Differentiating for $\boldsymbol{u}$, $\boldsymbol{x}_1$ and the Lagrange multipliers $\boldsymbol{\lambda}$ and then equating to $\boldsymbol{0}$ results in the constrained solution in matrix form:

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{x}_1 \\ \hat{\boldsymbol{\lambda}} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{S}_{\boldsymbol{u}}^\top \boldsymbol{Q}\boldsymbol{S}_{\boldsymbol{u}} + \boldsymbol{R} & \boldsymbol{S}_{\boldsymbol{u}}^\top \boldsymbol{Q}\boldsymbol{S}_{\boldsymbol{x}} & \boldsymbol{S}_{\boldsymbol{u}}^\top \boldsymbol{K}^\top \\ \boldsymbol{S}_{\boldsymbol{x}}^\top \boldsymbol{Q}\boldsymbol{S}_{\boldsymbol{u}} & \boldsymbol{S}_{\boldsymbol{x}}^\top \boldsymbol{Q}\boldsymbol{S}_{\boldsymbol{x}} & \boldsymbol{S}_{\boldsymbol{x}}^\top \boldsymbol{K}^\top \\ \boldsymbol{K}\boldsymbol{S}_{\boldsymbol{u}} & \boldsymbol{K}\boldsymbol{S}_{\boldsymbol{x}} & \boldsymbol{0} \end{bmatrix}^{-1}}_{\boldsymbol{\Sigma}_{\boldsymbol{u}}^{\boldsymbol{O}}} \begin{bmatrix} \boldsymbol{S}_{\boldsymbol{u}}^\top \boldsymbol{Q}\bar{\boldsymbol{x}} \\ \boldsymbol{S}_{\boldsymbol{x}}^\top \boldsymbol{Q}\bar{\boldsymbol{x}} \\ \boldsymbol{0} \end{bmatrix} \quad . \tag{5.16}$$

Dropping the third columns and rows corresponding to the equality constraint, results in a non periodic solution that produces an optimal initial state $\boldsymbol{x}_1$. This is useful to mimic the appearance and kinematics of trajectories that do not begin at a rest position, such as the case of a calligraphic movement that begins in the air rather than on the drawing surface. The extended solution (with or without the constraint) admits a stochastic sampling identical to the one defined above (Figure 5.7), which can be done by computing the eigen decomposition in equation (5.13) with the matrix $\boldsymbol{\Sigma}_{\boldsymbol{u}}^{\boldsymbol{O}}$ instead of $\boldsymbol{\Sigma}_{\boldsymbol{u}}$.

5.1.7 Multiple references

The output of the LQT optimisation procedure can be interpreted as a time varying flow field (Calinon and Lee, 2019) that depends on the minimisation of the tracking term of the cost

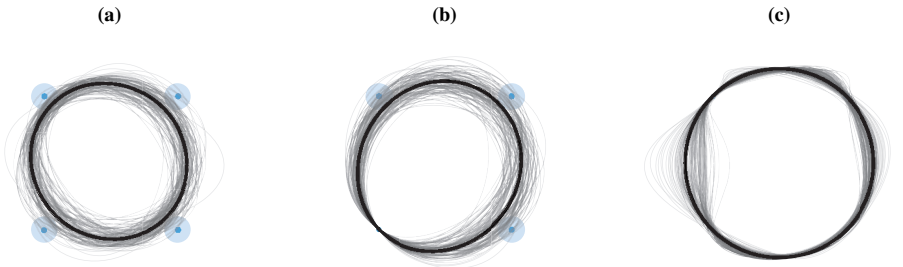


Figure 5.7: Stochastic sampling of periodic trajectories. **(a)** A sequence of Gaussians in a squared configuration, producing a circular trajectory and corresponding stochastic samples. **(b)** Increasing the precision of one Gaussian forces the trajectory and the sample to pass through its center. **(c)** Using a sparse tracking reference ($\sigma_h = 10^{-4}$) forces the trajectories to interpolate the centers, but allows for higher variance in the remaining trajectory segments.

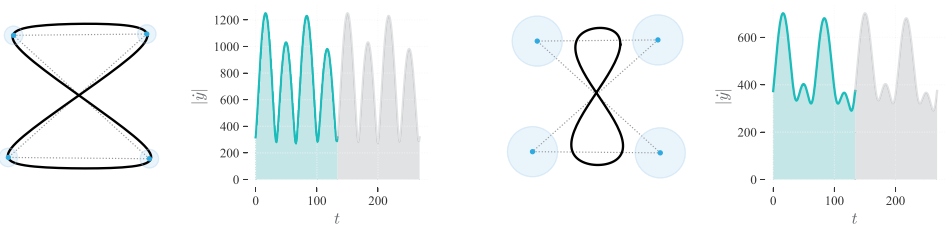


Figure 5.8: Periodic motions using Gaussians with different variances. The speed profiles are repeated (in light gray) to visualise the periodicity of the speed profile.

function. In addition to the definition of constraints, such a formulation can also be extended with additional quadratic costs that allow to track multiple references with different precision requirements. As an example, this is exploited in a robotic "learning by demonstration" application by Calinon (2016b), in which the cost function is expressed in a number p of different coordinate systems. The augmented cost function (from equation (5.9)) is given by:

$$J = \sum_{i=1}^{p} (\bar{\boldsymbol{x}}^i - \hat{\boldsymbol{x}})^\top \boldsymbol{Q}^i (\bar{\boldsymbol{x}}^i - \hat{\boldsymbol{x}}) + \boldsymbol{u}^\top \boldsymbol{R} \boldsymbol{u} \quad , \tag{5.17}$$

such that a trajectory of control commands can be retrieved with:

$$\boldsymbol{u} = \left(\sum_{i=1}^{p} \boldsymbol{S_u}^\top \boldsymbol{Q}^i \boldsymbol{S_u} + \boldsymbol{R} \right)^{-1} \sum_{i=1}^{p} \boldsymbol{S_u}^\top \boldsymbol{Q}^i \left(\bar{\boldsymbol{x}}^i - \boldsymbol{S_x} \boldsymbol{x}_1 \right) \quad . \tag{5.18}$$

Note that the multiple objectives are fully determined by the matrices $\boldsymbol{Q}^i$ and vectors $\bar{\boldsymbol{x}}^i$,

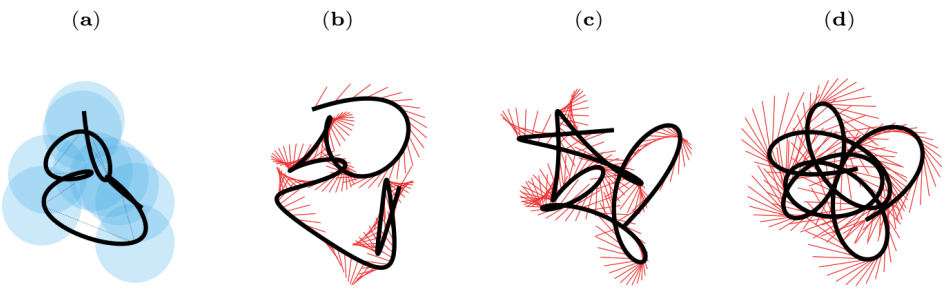


Figure 5.9: Adding a tracking reference with time varying velocity. **(a)** Trajectory with a single reference and isotropic covariance. **(b)** Adding a second reference to the objective, with a time varying (constant) velocity field (red segments) and starting from an initial orientation and a constant diagonal matrix $\boldsymbol{Q} = 5 \times 10^{-4} \boldsymbol{I}$. **(c)** Effect of varying the initial orientation. **(d)** Effect of increasing the diagonal term in $\boldsymbol{Q}$ to 1×10^{-3}.

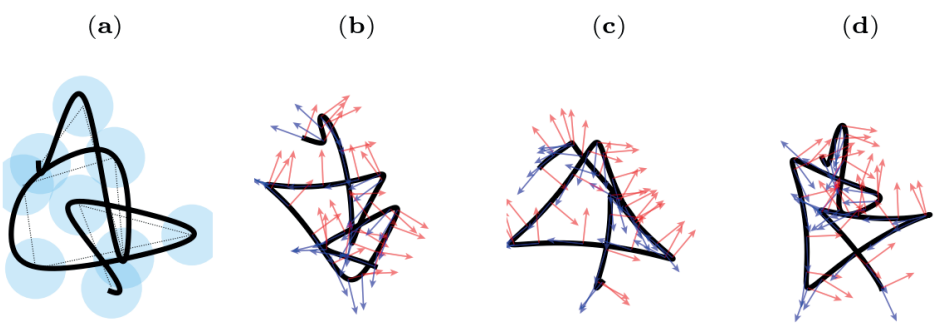


Figure 5.10: Additional reference with a time varying coordinate system on velocity. **(a)** Trajectory with a single reference and isotropic covariance. **(b,c,d)** Adding a second reference to the objective, consisting of a time varying coordinate frames (red and blue arrows) for the diagonal blocks of $\boldsymbol{Q}$ corresponding to velocity terms. This forces the the velocity at each time step to be parallel with the axes of the corresponding frame.

while the remaining terms remain constant across the sums. As a result the same procedure can be easily extended to the constrained case by replacing all the block entries of equation (5.16) in which the terms $\boldsymbol{Q}$ and $\bar{\boldsymbol{x}}$ appear, with the corresponding sums over the terms $\boldsymbol{Q}^i$ and $\bar{\boldsymbol{x}}^i$ for each reference.

Figure 5.9 shows an example application of this approach, where a trajectory is generated with one reference constructed with isotropic covariances (Figure 5.9.a). We then consider also a second reference ($p = 2$) consisting of a time varying velocity field. The reference

is constructed with a vector $\bar{\boldsymbol{x}}^2$ consisting of zeros everywhere except for the velocity terms form each time step. Each term consists of a vector that linearly changes orientation as a function of the time step. The corresponding weight matrix $\boldsymbol{Q}^2$ is zero everywhere with the exception of the diagonal entries corresponding to the velocity of each time step. These entries are set to a small value that determines a trade-off between the requirement to track the Gaussians, and the requirement to track the velocity specified in $\bar{\boldsymbol{x}}^2$. Using small weights and varying the orientation of the field results in different stylisations of the resulting trajectory (Figure 5.9b,c). A larger weight forces the trajectory to track the velocity and produces a result that is very different from the original (Figure 5.9).

With a slightly different approach (Figure 5.10), we can enforce a looser constraint that induces the trajectory velocities to be parallel with the axes of a rotated orthogonal coordinate system. This can be done by setting all entries of $\bar{\boldsymbol{x}}^2$ to zeros, and setting the diagonal blocks of $\boldsymbol{Q}^2$ to $\boldsymbol{v}_t \boldsymbol{v}_t^\top$, where $\boldsymbol{v}_t$ is the direction of one coordinate axis for a given time step. Both this method and the one explicitly using a time varying velocity reference can be used with higher derivatives as well (e.g. acceleration). These methods demonstrate a first example in which a user is able to simply specify a motor plan with an arbitrary number of vertices, and then to generate a variety of kinematic realisations of the motor plan by globally adjusting a few parameters. The parameters determine the global trajectory smoothness, as well as the way in which the second reference evolves in time (e.g. for this case, speed of rotation) and its weight (through a constant diagonal scaling of $\boldsymbol{Q}$).

5.2 User interfaces

The proposed trajectory generation method is efficient and is well suited for the interactive and procedural design applications. In an interactive setting, it is in fact trivial to drag the centers of the input Gaussians with a typical point-and-click procedure, and it is also easy to interactively manipulate the covariances. This can be done by manipulating an ellipsoid, where the center of the ellipsoid defines the mean $\boldsymbol{\mu}_i$, and the axes are used to manipulate the covariance $\boldsymbol{\Sigma}_i$ through its eigen decomposition:

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Theta}_i \boldsymbol{S}_i \boldsymbol{S}_i \boldsymbol{\Theta}_i^\top \quad , \tag{5.19}$$

where $\boldsymbol{\Theta}_i$ corresponds to an orthogonal (rotation) matrix, and $\boldsymbol{S}_i$ to a scaling matrix. For example, consider the 2D case in which the rotation and scaling matrices are given by:

$$\boldsymbol{\Theta}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}, \quad \theta_i = \tan^{-1}\frac{a_2}{a_1}, \quad \boldsymbol{S}_i = \begin{bmatrix} \frac{\|\boldsymbol{a}\|}{2} & 0 \\ 0 & \frac{\|\boldsymbol{b}\|}{2} \end{bmatrix} \quad , \tag{5.20}$$

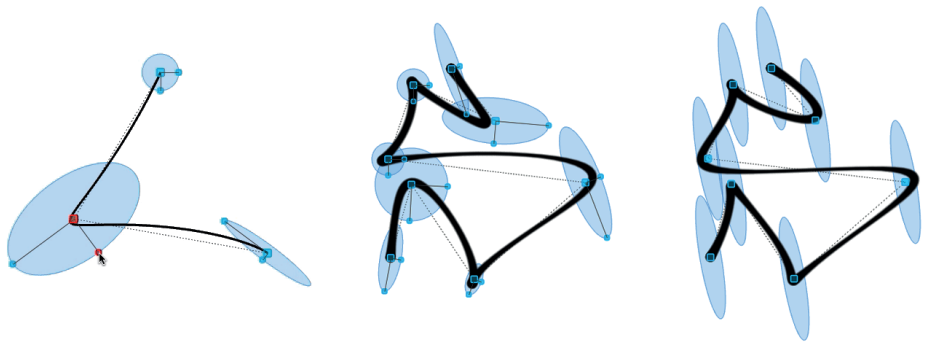


Figure 5.11: User interaction and kinematics driven brush rendering effects.

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are the major and minor axes of an ellipse, which can be interactively dragged to manipulate the shape of the distribution (Figure 5.11, left). While the examples given are two dimensional, an extension to three dimensional ellipsoids is straightforward to implement with a so called *arc-ball* interface (Shoemake, 1992).

Similarly to the case of the $\Sigma\Lambda$ model, MIC generates smooth trajectories that resemble the kinematics of a human hand motion. A trajectory generated over a sequence of M Gaussians will typically produce a speed profile characterised by $M-1$ peaks and local minima corresponding with curvature extrema along the trajectory, which is consistent with the stereotypical inverse speed/curvature relation seen in human movements (Lacquaniti et al., 1983). As a result, the exact same methods demonstrated in Chapter 4 can be used to generate and animate strokes by incremental sampling of a trajectory. One advantage of this method is that it easily generalises to dimensions higher than two, and additional coordinates can be used to smoothly vary parameters such as brush width (Figure 5.26).

5.2.1 Mimicking Bézier curves

The same optimisation framework can be used to mimic the shape and behavior of (cubic) Bézier curves, resulting in a user interface (UI) that is almost identical to its parametric counterpart. At the same time, this provides the flexibility of MIC, such as the ability to easily adjust the trajectory smoothness. Furthermore, MIC guarantees trajectory smoothness regardless of the configuration of control points. This is particularly useful for calligraphy generation, where the desired trajectories are usually smooth.

It has been shown that cubic Bézier curves (Egerstedt et al., 2004) and splines (Egerstedt and Martin, 2009) can be interpreted as the trajectories of a second order dynamical system which minimise acceleration commands. Indeed, we can see that with the previously described key-point formulation, it is possible to closely approximate a Bézier curve. This can be done by (i) setting the first order derivative entry of the sensor matrix $\boldsymbol{C}$ also to $\boldsymbol{I}$, then (ii) specifying a reference with Q_t zero for all time steps, except for the first and last ones

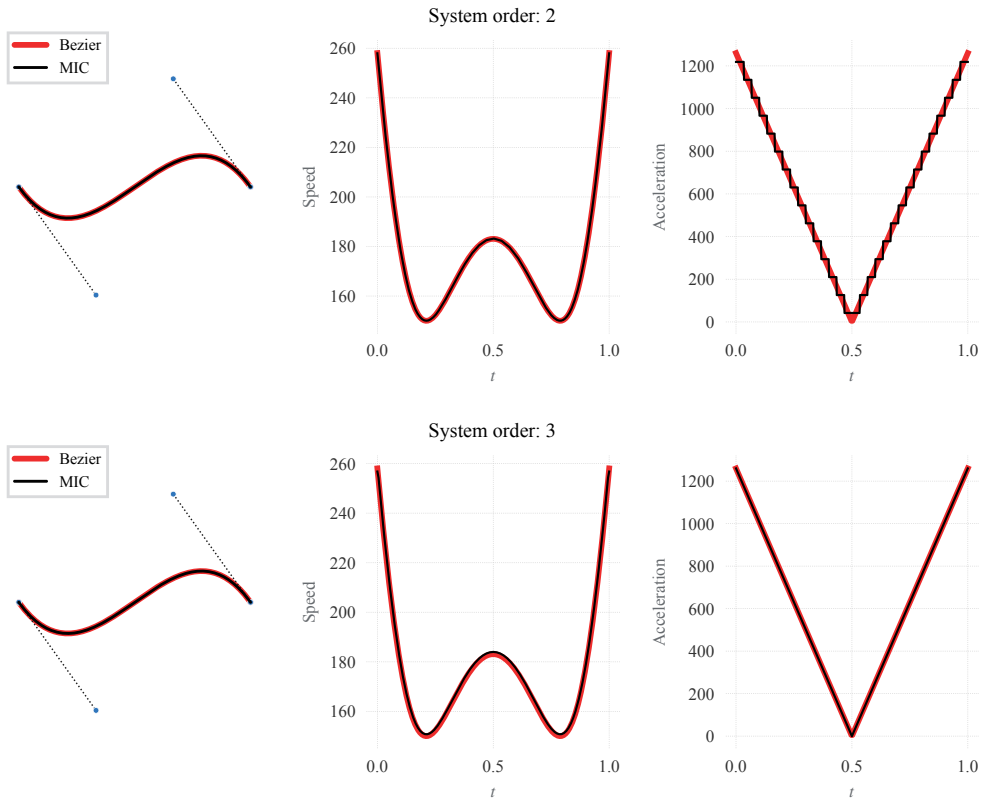


Figure 5.12: Approximating cubic Bézier curves (red) with optimal control (black). The curve control points are visualized as blue circles. First row, a 2nd order system computed with $N = 30$ timesteps and two input states consisting of the initial and final positions and velocities. As the number of timesteps increases, the resulting trajectory converges to that of a Bézier curve as shown by Egerstedt et al. (2004). Second row, the Bézier curve can also be reproduced by using a 3rd order system and computing the optimisation for only two time steps, while also optimising for the initial state $\boldsymbol{x}_1$.

that are set to $\boldsymbol{C}^\top \boldsymbol{C}$, and, finally (iii) specifying the corresponding desired states with the position and derivative of the first and last Bézier control points. This method allows to closely approximate a Bézier curve.

At the same time, we observe that we can also mimic the behavior and shape of a Bézier curve by using a step-wise tracking reference. This can be done by placing isotropic covariance Gaussians centered at each control point of the curve, and then adjusting the influence of intermediate control points on the trajectory by uniformly increasing the variance of each corresponding Gaussian (Figure 5.13).

If we relax the constraint of reproducing accurately the Bézier traces, we obtain a curve

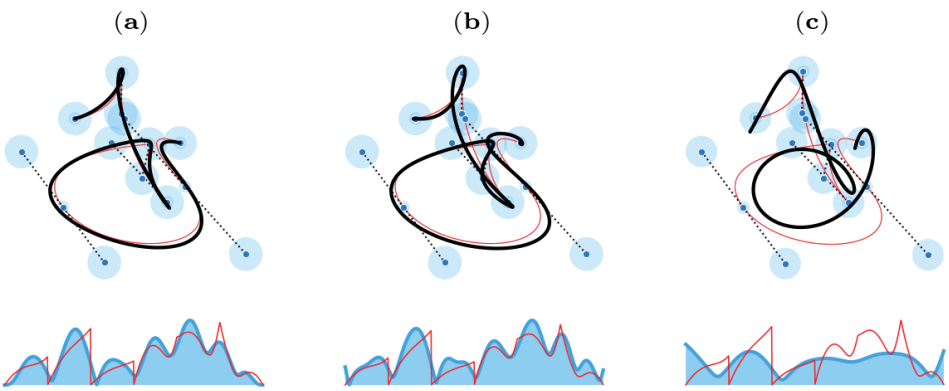


Figure 5.13: Mimicking piecewise cubic Bézier curves (created with Adobe Illustrator, in red) with MIC (black) using a stepwise reference $\sigma_h = 1$, isotropic Gaussians and a 4th order system. Below, the corresponding speed profiles normalised and superposed for comparison. Note that the Bézier speed (red) goes to zero because Illustrator produces curve segments with coincident control points. **(a)** Initial trajectory resulting from Bézier control points. **(b)** solving also for the initial state $\boldsymbol{x}_1$ results in a slightly different trajectory that does not begin and end at a rest position. **(c)** Decreasing the maximum displacement parameter produces a global smoothing effect.

generation method that produces similar curves, but with the additional flexibility of the Gaussian representation and the benefit of always maintaining smooth and physiologically plausible kinematics. The utility of this property in our application is emphasized if we randomly perturb the control points of a letterform and compare the result with the one produced with a Bézier curve (Figure 5.14). In the examples given, the variances have been set empirically, but the results suggest that it should be possible to identify a more systematic relation that leads to an optimal reproduction of the Bézier curve, while guaranteeing trajectory smoothness. This is left as an avenue for future research.

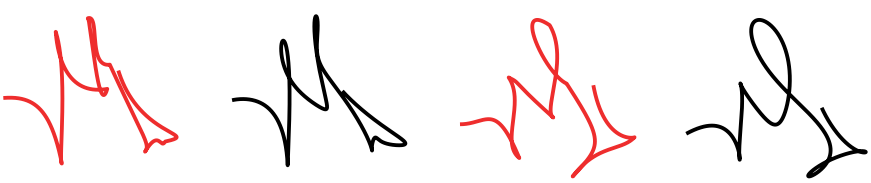


Figure 5.14: Effect of randomly displacing control point positions with Bézier curves (red) and MPC (black).

While the Bézier curve becomes discontinuous due to the differently oriented tangents at the loci where the curve segments meet, the MPC formulation tends to maintain a smooth

trajectory regardless of the positions of the control points. This can be exploited as an additional method to generate synthetic variations of a handwriting or calligraphy trajectory, which can be interactively edited with a traditional control point and tangent interface. The same smoothness property can be used to concatenate multiple letterforms with ligatures that evoke a smooth and natural motion, which can easily be achieved by treating the control points of the letters as a single trajectory (Figure 5.15).

5.2.2 Semi-tied structure

In the previous sections, we have seen that it is possible for a user to easily generate variations of MIC trajectory in 2D applications by editing the position of bi-variate Gaussians. For applications aimed at procedural content generation, it may be desirable to formulate a more parsimonious way of generating trajectories, in which different stylisations are generated without having to specify the covariance of each MoG component.

We observe that one convenient way to achieve this result is to enforce a *shared orientation* for all covariance ellipsoids. This is easily achieved with the formulation above by keeping the orientations $\mathbf{\Theta}_i$ to a fixed value and results in a "semi-tied" covariance structure (Tanwani and Calinon, 2016) of the input MoG, in which all covariances share the same eigenvectors but not necessarily the same eigenvalues. From a motor control perspective, the semi-tied formalism can be interpreted as the alignment of different movement parts/primitives with a shared coordination pattern (Tanwani and Calinon, 2016), which is in line with the hypothesis of postural-synergies at the motor planning level (d'Avella et al., 2003). This implies a shared non-orthogonal (oblique) basis for all the covariances, which produces a shear transformation that in the 2D case transforms a circle into an oriented ellipse (Figure 5.17). Oblique coordinates have been suggested to describe the coordination of handwriting movements made with the fingers and wrist (Dooijes, 1983), which suggests another possible biophysical interpretation of this result.

Semi-tied covariances provide a simplified parameterisation that allows to explore different stylisations of a key-point sequence with a reduced set of open parameters. The semi-tied covariances enforce a coupling between the coordinates of the trajectory, which results in a sense of coordination in the movement. At the same time, minimization of the control

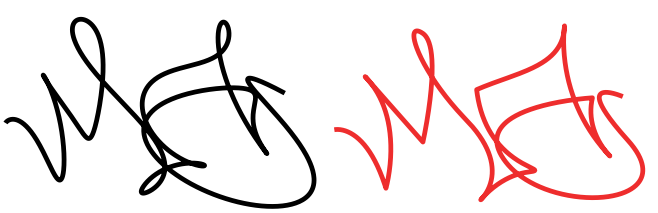


Figure 5.15: Automatic ligature generation by concatenating the control points of two letters. On the right, a comparative example using Bézier curves.

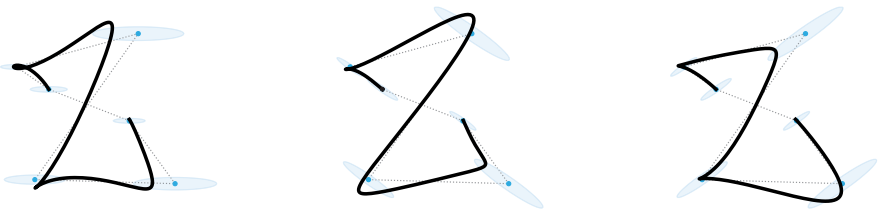


Figure 5.16: Different stylisations of a letter "Z" using semi-tied covariances with different orientations.

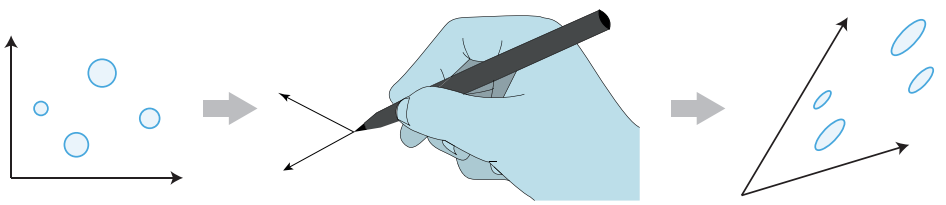


Figure 5.17: Illustrative example of the oblique coordinate system that could result from the fine movements in handwriting made by rotating the wrist on a fixed point.

command amplitude produces smooth trajectories that evoke a natural drawing movement.

It is then easy to edit the semi-tied covariances with an interface in which the user can drag the basis vectors of $\boldsymbol{H}$ and scale the value of h (Figure 5.18). Because the cost function used in the optimisation is given by a tradeoff between tracking and control costs, it is possible to keep the maximum displacement $\Delta_{\max}$ (which determines the control weight) to a fixed value proportional to the workspace area. The user can then define the smoothness of the generated trajectory by manipulating h, where an increase in h produces larger covariances and consequently smoother trajectories.

5.3 Calligraphic stylisation

With the previously described interfaces, a user can interactively explore different stylisations of a target sequence. While the semi-tied covariances enforce a sense of coordination in the movement, the minimisation of the control cost produces smooth trajectories that evoke a natural drawing movement (Figure 5.19). A similar method can also be used to generate different stylisations of an input trace, by placing motor plan vertices near its curvature extrema, which we explore next.

5.3.1 Reconstructing instances of calligraphy

The previous approach can be used to rapidly reconstruct and generate variations of an existing instance of human made calligraphy (Figure 5.20) or graffiti (Figure 5.21). In such appli-

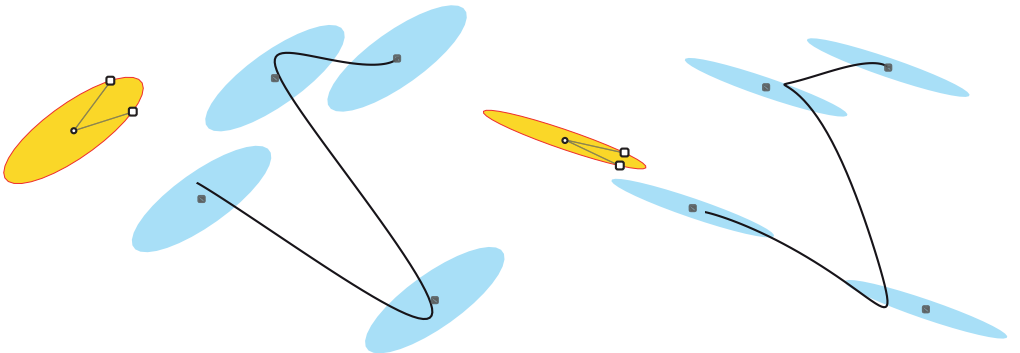


Figure 5.18: Interface for manipulating semi-tied covariances and corresponding trajectories. The user can drag at the border of the yellow ellipsoid the pair of small white rectangles to redefine the basis vectors of $\boldsymbol{H}$ with magnitude h which directly impacts all covariances at once.

cations, the user first defines a motor plan with a coarse sequence of Gaussians over salient positions along the input trace (approximately in correspondence with perceived curvature extrema), and then adjusts the covariances to modify the trajectory and mimic the curvature and smoothness of the original trace.[3] Different kinematic realisations and stylisations of the input can then be generated by either globally varying the covariances or by using stochastic sampling. We will demonstrate the usefulness of this approach with automatically determined motor plans in Chapter 11.

5.3.2 Predefined motor plans

We also test calligraphic stylisation with combinations of user-defined motor plans meant for letterforms. To limit the effect on stylisation of the letter structure we use prototypical glyphs that are adopted to teach print letter writing to children (Zaner-Bloser method, 2020), and set the vertices approximately in correspondence with points along the glyph traces at which the horizontal or vertical direction of the pen would change (Figure 5.22.a). We then assign

[3] We note that with our UI this proves easy to do.

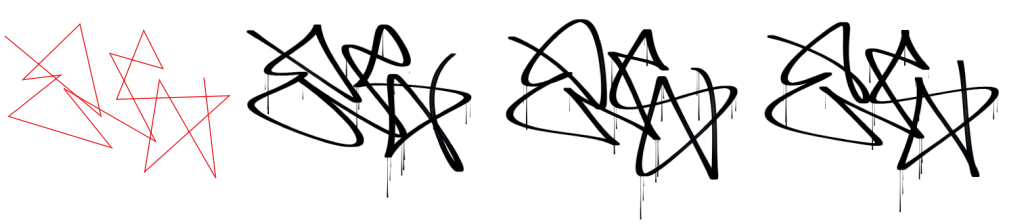


Figure 5.19: Calligraphic stylisations of a user-defined motor plan (red). The stylisations are generated with the brush model described in Chapter 4 and by varying the global orientation of semi-tied Gaussians and scaling their variance.

Figure 5.20: Reconstruction of an instance of calligraphy by New York artist David Chang (courtesy of the artist) with our interactive user interface.

a Gaussian with the same covariance to each target, producing a trajectory that we consider to be evocative of stylised handwriting (Figure 5.22.b). We increase the degree of stylisation by adding vertices to each spine so that the movement begins with either a down-stroke or a stroke that moves from left to right (Figure 5.22.c). Furthermore, we observe that by concatenating the motor plans of consecutive letters into a single one (Figure 5.22.d) results in smooth ligatures that are evocative of natural writing motion. We test a similar procedure on more abstract and complex letter templates (Figure 5.23), which results in an accentuated and diverse visual effect of the different stylisation parameters.

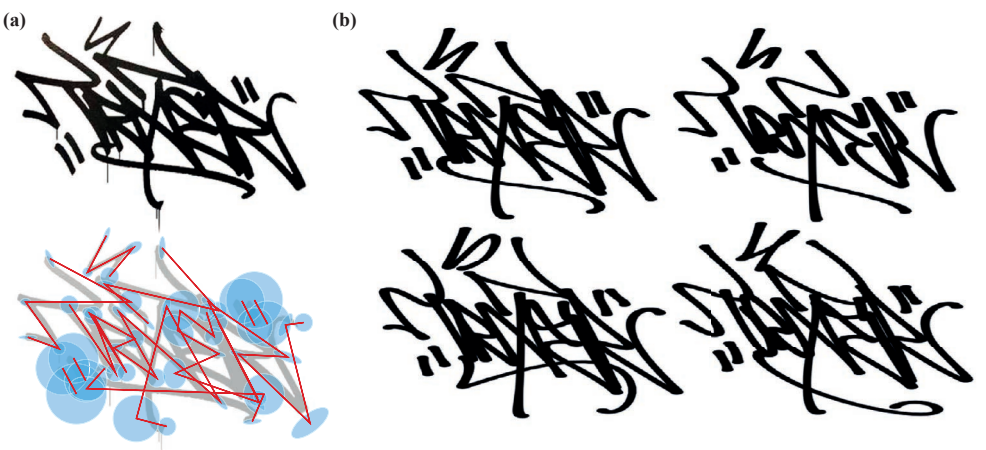


Figure 5.21: User reconstruction and variation of a graffiti tag. **(a)** Top left: a graffiti script (tag) made with a marker by Los Angeles artist "Trixter" (courtesy of the artist). Bottom left: user defined motor plan and Gaussians for the tag (above), generated by placing points near salient positions along the original trace and then manually adjusting the covariances to follow the original trajectory. **(b)** Top row: the reconstructed trajectory and one variation made by increasing the regularisation parameter r. Bottom row: two random samples from the trajectory distribution of the reconstruction.

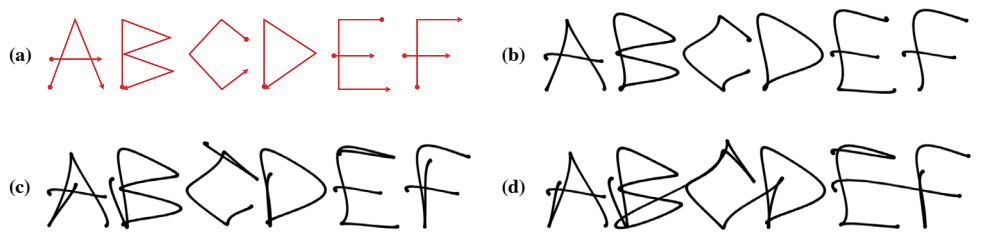


Figure 5.22: Stylisation of simple alphabet letters. **(a)** Target sequences for the letters. **(b)** Trajectories generated with MIC (using covariances oriented with $\theta = 116^\circ$). **(c)** The same trajectories with the addition of down strokes and left-to-right strokes. **(d)** Addition of some ligatures between letters.

5.3.3 Generating Asemic Tags

We have seen how the probabilistic formulation of MIC together with a semi-tied covariance formalism can be used to rapidly explore different stylisations of a letter structure, defined

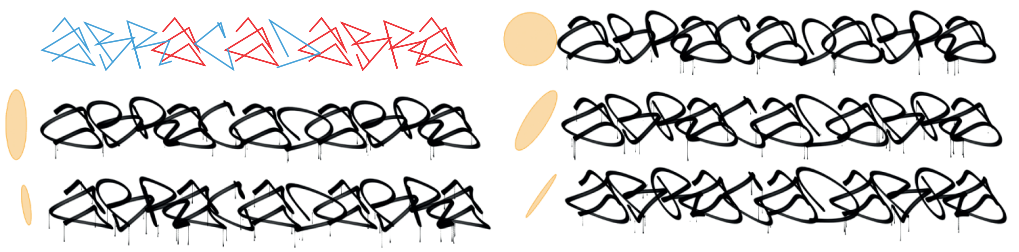


Figure 5.23: Composition of predefined motor plans for the letters "ABRCD" (blue) composing the word "ABRACADABRA" and stylised by repeating the same covariance for each motor plan vertex. Next to each stylisation the repeated covariance ellipse in yellow.

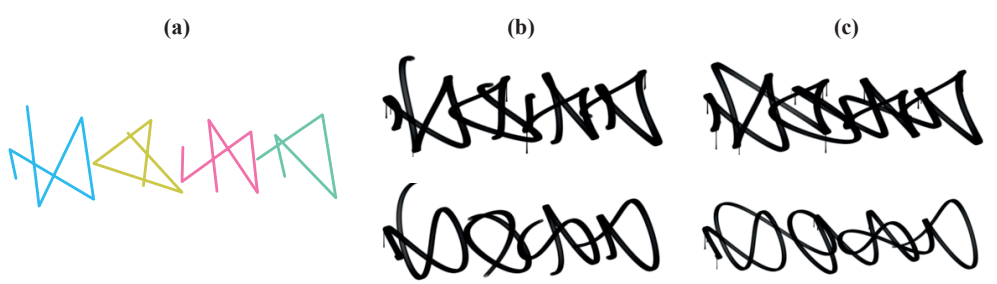


Figure 5.24: Asemic tags. **(a)** A sequence of four asemic glyphs. **(b,c)** Different calligraphic stylisations using semi-tied covariances and mimicking graffiti tags. Each row is generated with the same parameters (Gaussian orientation, covariance, $\Delta_{\max}$), but the instances in the column **(c)** are generated by concatenating all the glyphs as a single motor plan (i.e. generating ligatures between asemic forms).

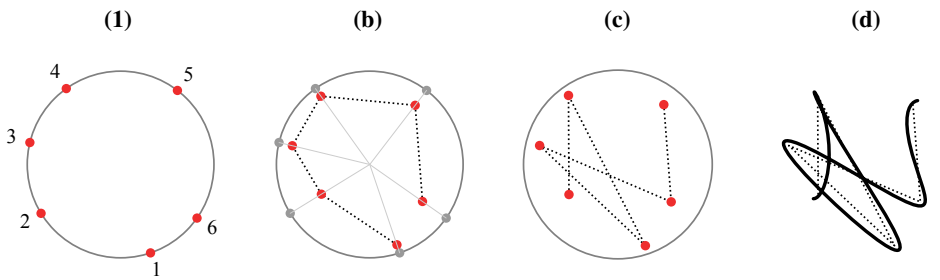


Figure 5.25: Asemic glyph generation procedure.

as a coarse sequence of targets. This parsimonious representation can be exploited in combination with procedural generation methods. The user is then left with the simplified task of generating coarse point sequences, while the stylised trajectory evolution is generated by optimal control.

Here we demonstrate a simple application, in which glyph-like structures with no intended semantics, that is "asemic letters" (Figure 5.24.a), are generated procedurally and then rendered with different styles by optimal control (Figure 5.24.b and c). The procedure to generate a random asemic glyph is simple and consists of 3 steps (Figure 5.25):

(a) Generate an ordered sequence of m points randomly distributed along a circle.

(b) Offset each point by a random amount, along the radial vector from the circle center.

(c) Refine the ordering of the points so the sequence maximises the distance between consecutive points, alike an "inverse" travelling salesman problem (TSP), and rewards certain stroke directions that might facilitate motor execution by a drawing hand (e.g. down and left-to-right strokes).

.

5.3.4 Stroke thickness

The same optimisation procedure can be used to smoothly vary the thickness of a brush, by considering its radius as an additional coordinate in the reference trajectory. To do so, we add a user configurable diagonal entry to the input covariance matrices which determines the allowed variability and smoothness of the radius. The result is similar to the output of a disk B-spline (Seah et al., 2005), but with control on the smoothness of the stroke thickness profile (Figure 5.26.b,c). Because trajectory samples are denser where the speed is lower, using this method with a transparent brush results in an effect that mimics a greater deposition of ink or paint near curvature extrema (Figure 5.26.d).

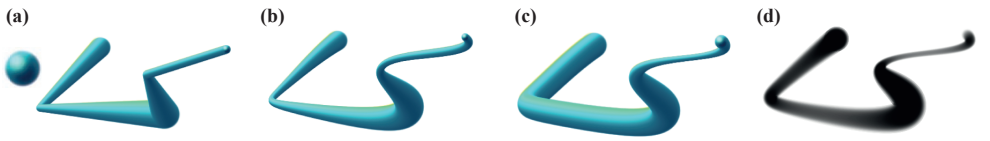


Figure 5.26: Variable brush thickness smoothing. **(a)** The texture on the left is swept along the segments of a motor plan with a thickness that varies linearly between vertices. **(b)** Sweeping the same texture along a smooth trajectory that also tracks the brush thickness defined at each vertex. **(c)** Effect of increasing the variance for the thickness coordinate. **(d)** Using a semi-transparent brush results in denser brush samples near curvature extrema.

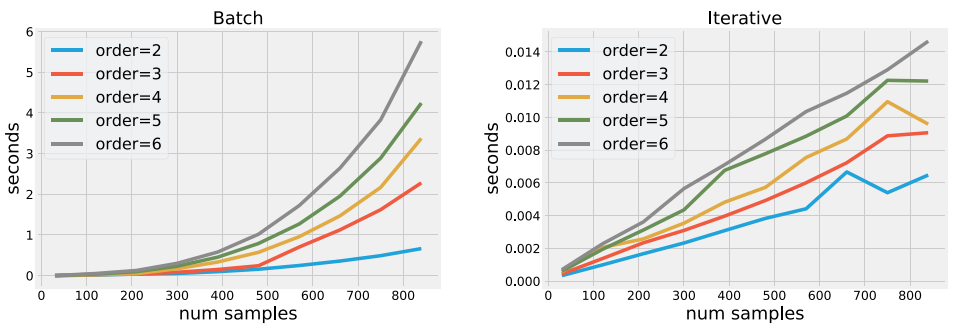


Figure 5.27: Comparison of performances between the batch and iterative approaches.

5.4 Discussion

5.4.1 Performance

We have tested our method on a 2.7 GHz Intel Core i7 machine and used OpenGL for hardware accelerated rendering; We have implemented the optimisation code in Python, using the NumPy (Van Der Walt et al., 2011) linear algebra package, as well as in C, using the Armadillo library (Sanderson, 2010). Both the batch and the iterative approach (discussed in section C.2) run at interactive rates up to a limit of time steps that depends on the order of the system and on the sparsity of the tracking reference used in the optimisation (Figure 5.27). The batch solution requires the solution of a linear system of equations which can be done with a time complexity of $O(n^3)$ and becomes rapidly non-interactive as the number of samples increases. However we observe that a sparse sampling, e.g. $T_s = 5$ produces the desired trajectory behavior determined by the Gaussians, and results in a problem that is manageable also for complex trajectories.

5.4.2 Limitations: passage times

One limitation of the proposed method relates to our assumption of perfect isochrony and uniformly spaced passage times. As can be seen in Figure 5.28, this assumption results in trajectories with curvature extrema that do not always coincide with interpolation points or

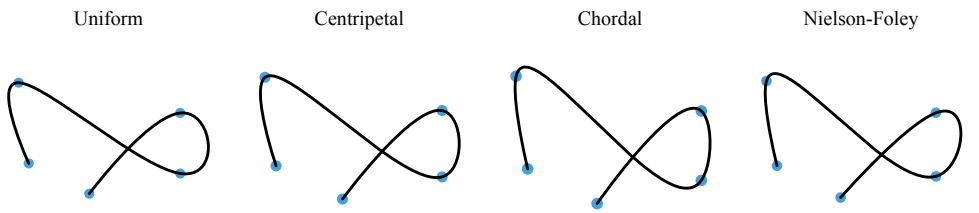


Figure 5.28: Cubic interpolation with MIC using a second order system and different parameterisations: Uniform (default), centripetal, chordal and Nielson-Foley. The interpolatory behavior is produced with a value σ_h close to zero and forcing zero velocity and acceleration for the last state.

locations that are perceptually related to the location of a Gaussian center. This issue is identical to the one with uniform spline parameterisation, which is well known for the case of cubic curves used in the CAGD domain, and has resulted in the proposal of methods such as Catmull-Rom splines, or non-uniform parameterisations such as centripetal (Lee, 1989), chordal (Floater and Surazhsky, 2006) and Foley-Nielson (Foley and Nielson, 1989). For the case of densely sampled trajectories, non uniform parameterisations can easily be adapted to our method, by appropriately choosing the passage times (Figure 5.28). For sparse samplings, this requires formulating the optimisation with time-varying transfer matrices, which we leave as an avenue for future developments.

It should be also noted, that while our method is consistent with the motor control hypothesis of minimum squared derivatives, such methods do not predict a perfect local isochrony, but rather passage times that are *approximately* uniform across a given movement (Figure 5.29a). As an example, the passage times for the minimum jerk (MJ) model are predicted as a byproduct of the optimisation, and this results in the desirable property that the resulting trajectories interpolate via-points almost exactly at curvature extrema. However, the solution for time of trajectories with more than one via-point requires a non-linear optimisation procedure that cannot be performed in real time, at least with the methods described in the literature (Figure 5.29b).

It is interesting to note that applying the centripetal parameterisation to a trajectory generated with a third order system results in a more precise approximation of the optimal MJ trajectory (Figure 5.29c). As the name implies, the centripetal parameterisation is based on a low order approximation of centripetal acceleration. This result suggests that a similar approximation heuristic can be developed for higher order derivatives, which could be a beneficial extension to our method.

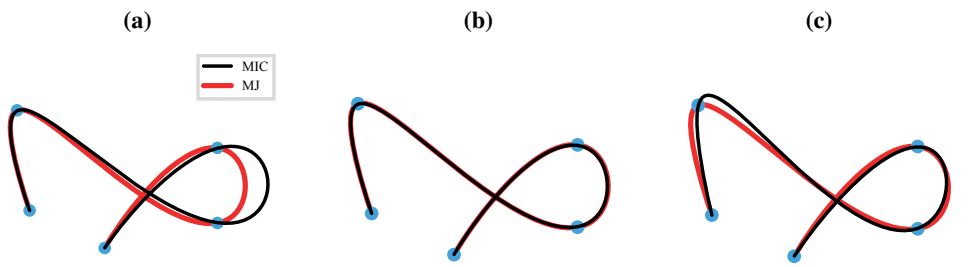


Figure 5.29: Comparing an MJ trajectory (red) and an interpolating trajectory generated with MIC using a third order system (black). **(a)** Comparison with uniform parameterisation. **(b)** Comparison with passage times computed using the method of Todorov and Jordan (1998), resulting in identical trajectories. **(c)** Comparison with passage times computed with the centripetal method of Lee (1989).

5.5 Conclusion

We have presented MIC, a method for the generation of smooth curves and motion trajectories using a stochastic solution to an optimal control problem. The output of our method is a trajectory distribution, which describes a family of motion paths that can mimic the appearance and the variability of human-made artistic traces. The input to the method is a sparse sequence of multivariate Gaussians that determine the overall shape of the output and explicitly define its variability. This results in a representation that is similar to the one used in conventional CAGD applications, and that can be edited interactively in a similar manner. While in this chapter we have focused on the generation of 2D trajectories, the proposed methodology can be generalised to higher dimensions. This opens up the possibility to extend the method to 3D trajectories, as well as taking into consideration the evolution of additional variables, such as the drawing tool orientation, pressure or color.

For our use case, we let the user explicitly define the MoG components. However a similar representation can be learned from data with standard maximum-likelihood estimation methods (Calinon, 2016a). Our choice of Gaussians as an input and output distribution is principally motivated by its effectiveness and simplicity of representation. From a user-interaction perspective, this allows users to intuitively manipulate the input distributions by modifying the axes of each MoG component ellipsoid (Figure 5.11). Furthermore, the straightforward relation of Gaussians to quadratic error terms in linear systems allows us to solve the optimal control problem at interactive rates and in closed form, all the while offering a stochastic interpretation of the output. Extending the proposed method to non-linear dynamical systems and to distributions other than Gaussians represents an interesting av-

enue of future research.

Similarly to the previously described ΣΛ model in Chapter 4, each trajectory generated by the MIC method reflects a movement with physiologically plausible kinematics. This can be exploited to produce appealing rendering effects, realistic (to the human eye) animations, or even to generate smooth motions that can be tracked with a robotic arm (Berio et al., 2016). One advantage of this method with respect to the ΣΛ model, is that it allows to produce consistent variations and stylisations of a trajectory with a very compact parameter set, for example with the use of semi-tied covariances or the addition of multiple tracking references. We exploit this property in the next chapter, where we use MIC trajectories to generate the outlines of solid strokes similar to the ones that can be seen in graffiti "pieces".

While less flexible to control parametrically, we will take advantage of the explicit sub-movement parameterisation given by the ΣΛ model in Chapter 8, where it will be used as a basis for a data-driven approach to generate calligraphic stylisations from examples.

Chapter 6

Outline stylisation: Sketching and layering thick graffiti primitives

"The arrow... everybody's got their own arrow. I like that though."

Zephyr, Style Wars

Figure 6.1: Examples of outputs from our system. **(a)** Stylistic variations of the same stroke describing a letter "S". **(b)** A letter "S" with local layering and arrows at the stroke ends. **(c)** Combination and layering of multiple strokes and rendering effects for the graffiti composition "EXPRESS".

This chapter is based on a collaboration established by myself and Prof. Frederic Fol Leymarie together with Dr. Paul Asente and Dr. Jose Echevarria at Adobe Research (San Jose, California). All the methods have been developed and implemented by myself, with the ex-

ception of the 2D extrusion method described in Section 6.3, which has been developed by Paul Asente as an unreleased plugin for Adobe Illustrator. An earlier version of the work reported here is published (Berio et al., 2019).

The trajectory generation methods developed in the previous two chapters result in a system that allows to interactively or procedurally generate strokes that resemble the ones seen in calligraphy or graffiti tags. These types of strokes are inherently 1D, and mimic the rapid traces left by the motion of a writing or drawing instrument. Similarly to typography, in more sophisticated forms of graffiti art, strokes take on a 2D form and are combined to produce an outline. Depending on the graffiti sub-genre, these strokes are either sketched with skillful free hand motions or precisely traced in a geometric way. Strokes are often interlocked in complex ways and may have self-occlusions and loops (Ferri, 2016). They are then fused and traced to create the outline of a highly stylised version of one or a combination of letterforms (Figure 6.1). The resulting outlines are not limited to the boundary of the letter, but may extend to suggest where and how different strokes overlap or where a stroke folds over itself. The result is often evocative of a 3D composition.

In this chapter, we follow a similar strategy to the previous two, and develop an interface that is inspired by the way in which graffiti pieces are typically composed. While each artist's method is idiosyncratic, common construction strategies exist (e.g. (Schmidlapp, 1996, p.61)). First, a letterform is conceived as motor plan, along which stylised strokes and parts are then combined in a rough sketch (Arte, 2015).[1] Second, the union of these elements is filled in, often with a combination of gradients and geometric forms that follow the overall letter and outline structure. Third, and finally, this union of letterforms is outlined to reveal an overall view consisting of possibly interlocking and overlapping parts. Effects such as extrusions and highlights are often furthermore added.

Reproducing such results with conventional vector drawing digital tools can be very challenging. Many, if not all, well known such software packages assume that objects are separately layered in a back-to-front order (Adobe, 2019b). As a result, creating interlocking patterns and overlaps requires either manually masking hidden parts of an outline, or cutting overlapping parts and manually removing occluding parts of object outlines. Other applications, like Adobe Animate (Adobe, 2019a), support planar map decompositions , but do this in a way that does not maintain the continuity of the original strokes. In both cases, the required manual interactions are time consuming and, more importantly, one loses the underlying structure of the drawing. This makes it difficult to perform changes and explore variations of a drawing.

[1] With expertise, multiple parts may be sketched as a whole, which can lead to more sophisticated and organic forms. However, the various parts are often conceived as independent stroke-like elements.

As a solution to this challenge, we propose an interactive computational model of "graffiti strokes" (Section 6.1) and develop a method for rapidly combining such strokes into letters and other interlocking patterns (Section 6.2.3). Our stroke model relies on a variant of the popular *skeletal strokes* technique (Hsu and Lee, 1994) that we extend to be able to mimic the appearance of artful complex graffiti (Figure 6.2). We exploit the stroke structure to develop an efficient method and interface for handling complex layering and self-overlaps. The final output of our method is a set of non intersecting outlines, like the ones produced by hidden line removal methods in 3D, but relying on a fully 2D representation and interface.

6.1 Stroke Generation

The basis for our stroke generation method is a variant of the popular skeletal strokes technique (Hsu and Lee, 1994). We recall from Chapter 3 that a skeletal stroke is defined as an input shape, called a *prototype*, that is deformed along a destination path, called *spine*. The deformation is performed by mapping portions of the prototype to portions of the spine, and then generating outlines using a variable-width profile.

Width profile. Typical skeletal stroke implementations assume that the width varies continuously along the spine. However, we observe that components of graffiti letters often have widths that change discontinuously at spine corners, resulting in an effect that evokes a 3D projection of a surface or the trace of a chiseled calligraphic pen. To facilitate this, we define a spine as a sequence of vertex pairs, where each vertex pair is connected by a *segment* and each segment has an initial and final width (Figure 6.3).

The subsequently developed stroke model uses only straight segments, but the method is general enough to work with curved segments as well.[2]

Prototype deformation. In the standard case of a continuous width profile, the prototype can be deformed by mapping points along its outline to points that are perpendicular to the spine. These points are given by a sequence of line segments centered along the spine called *ribs*, which are orthogonal to the spine and have lengths depending on the width profile. This approach can lead to self-folds in the deformed prototype, corresponding to corners and high-curvature portions of the spine that produce retrograde motion in the resulting stroke outline (Asente et al., 2007). Such folds are often considered undesirable, and the usual approach is to avoid them by adjusting the orientations and lengths of the ribs. This can be done globally (Hsu and Lee, 1994), by using the angle bisectors rather than the normals at corners and interpolating the intermediate ribs accordingly. Another approach is to perform the adjustment locally (Asente et al., 2007), which avoids the potentially skewed appearance of the strokes.

[2] Both polygonal and curved segmented spines have been considered in my implementations and have been tested with success.

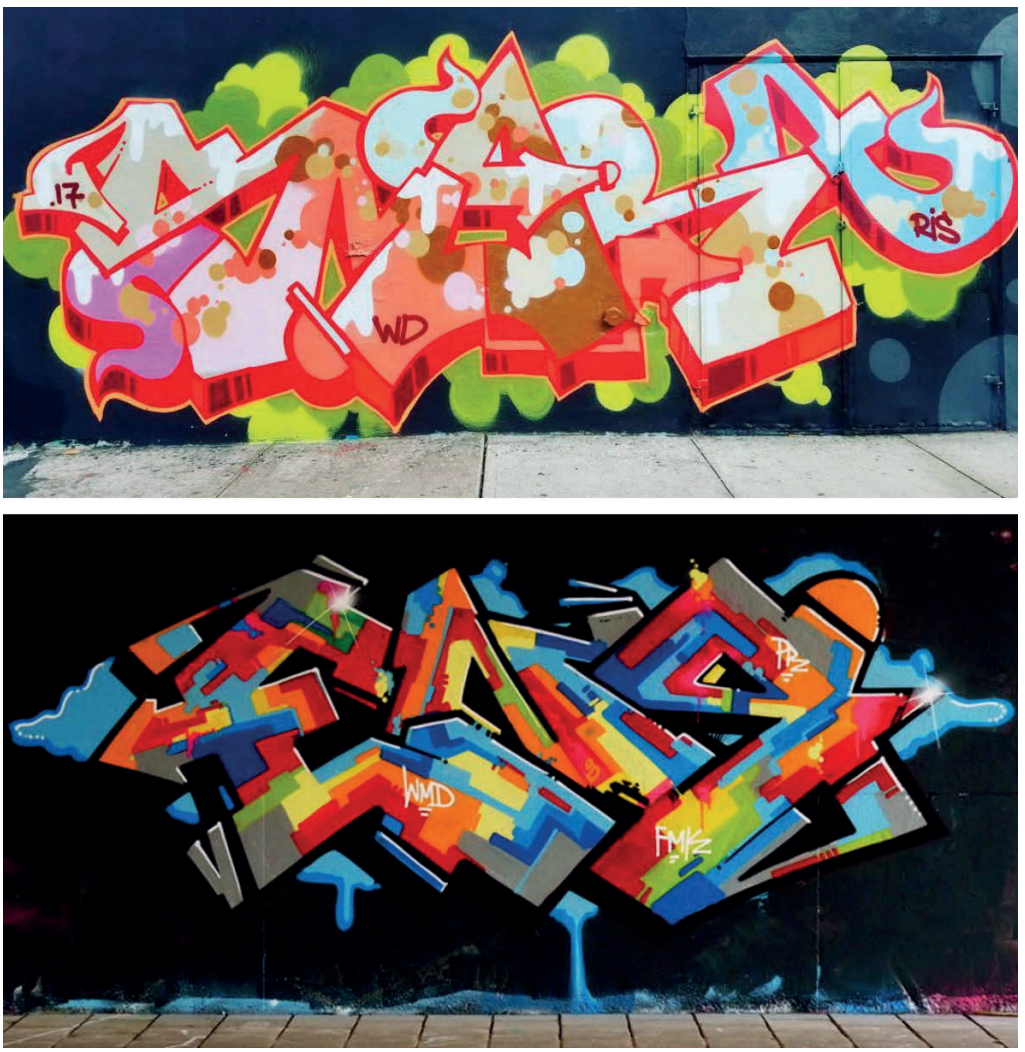


Figure 6.2: Graffiti with complicated intertwined strokes, courtesy of the graffiti artists SMART (top) and ENS (bottom).

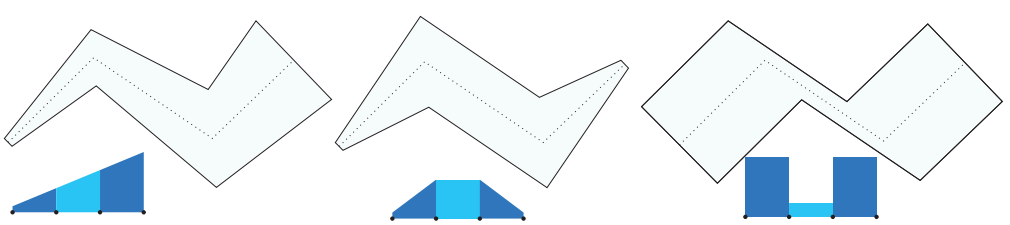


Figure 6.3: Strokes with rectangular prototypes and varying width profiles (shown below). Each color represents a different segment.

For the desired use-case, we must accommodate width discontinuities at spine corners. Moreover, many graffiti styles use folds instead of avoiding these (Figure 6.1). Our definition of ribs thus differs slightly from the one used by Hsu and Lee (1994) and others.

In our case, each spine segment is covered by a series of ribs that interpolate an initial and a final rib defined at the segment end-vertices. The first and the last spine vertices are assigned one rib each, and the rib is perpendicular to the incident spine segment. The intermediate spine vertices are assigned two ribs each, one for each incident spine segment. The orientations and lengths of these two ribs are computed with a procedure that is discussed next, and which depends on the width profile and on the orientation of the incident spine segments.

The ribs at each intermediate vertex (where the spine flexes) are defined using an oblique coordinate system $[\hat{\boldsymbol{u}}_1, \hat{\boldsymbol{u}}_2]$, centered at the vertex (Figure 6.4a), and given by:

$$\hat{\boldsymbol{u}}_1 = \hat{\boldsymbol{d}}_1 \operatorname{sgn}(\alpha) \quad , \quad \hat{\boldsymbol{u}}_2 = -\hat{\boldsymbol{d}}_2 \operatorname{sgn}(\alpha) \quad , \tag{6.1}$$

where $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$ denote the unit tangents preceding and following the vertex, α is the angle between the two tangents, and $\operatorname{sgn}(\alpha)$ ensures that the coordinate system is always oriented towards the *convex* part of the stroke. The offsets with respect to this basis are then given by:

$$o_1 = \frac{w_2}{\sin\alpha} \quad , \quad o_2 = \frac{w_1}{\sin\alpha} \quad , \tag{6.2}$$

with w_1 and w_2 denoting the profile widths preceding and following the vertex. This construction results in a *weighted bisector* $\boldsymbol{b} = o_1\hat{\boldsymbol{u}}_1 + o_2\hat{\boldsymbol{u}}_2$, whose direction is the same as the angle bisector when the widths on each side of the vertex are equal.

On the convex side of a vertex, we test the outline angle at $\boldsymbol{b}$ and generate a miter joint if it is too acute, as is done in conventional stroking algorithms. The ribs at the vertex terminate either at $\boldsymbol{b}$ or at the miter intersections. On the concave side, the ribs could end at the tip of the vector $-\boldsymbol{b}$, removing folds (Figure 6.4a) in a manner similar to the bisector-based method proposed in the original skeletal strokes implementation (Hsu and Lee, 1994). However, for our application we exploit the folds in order to render overlapping effects. To do so, we end the ribs at the tips of the vectors $\check{w}\hat{\boldsymbol{u}}_1 - \boldsymbol{b}$ and $\check{w}\hat{\boldsymbol{u}}_2 - \boldsymbol{b}$ (Figure 6.4b). Here $\check{w}$ is the minimum of o_1 and o_2 scaled by an angle fall-off function:

$$1 - \exp\left(-\frac{\alpha^2}{\sigma_\alpha^2}\right) \quad , \tag{6.3}$$

that decreases the amount of folding proportionally to the angle between spine segments, according to a user configurable parameter σ_α that we set experimentally to $\pi/4$ (Figure 6.5). This avoids excessive folding for obtuse angles and, for our use case, improves the visual

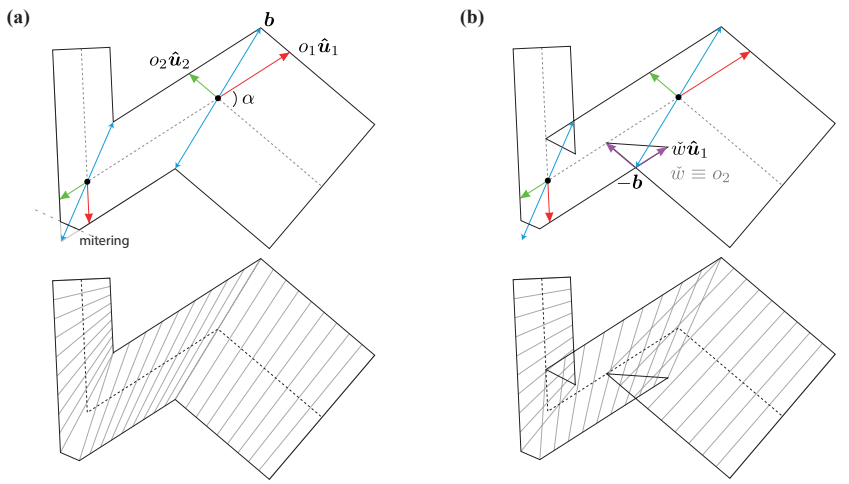


Figure 6.4: Corner rib adjustment according to the oblique coordinate system $[\hat{\boldsymbol{u}}_1, \hat{\boldsymbol{u}}_2]$ and corner mitering. **(a)** Unfolded construction similar to the one proposed by Hsu and Lee (1994). **(b)** Folded construction. Below, the ribs generated by each construction.

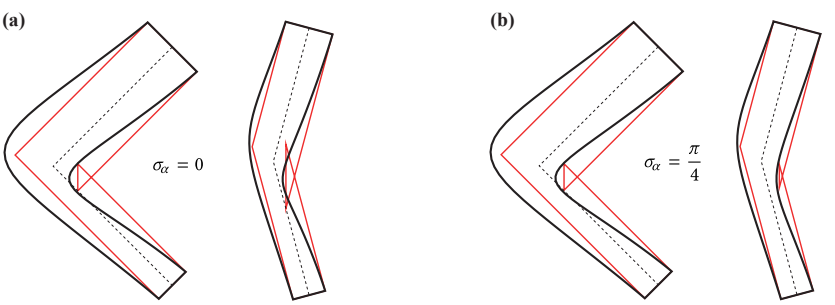


Figure 6.5: Effect of the angle fall-off parameter. When $\sigma_\alpha = 0$ the parameter has no effect and **(a)** shows the resulting strokes with different angles α between spine segments. A larger value of $\sigma_\alpha = \pi/4$ in **(b)** reduces the amount of folding proportionally to the angle α. This affects the resulting smoothed trajectory (black) only when the angle α is obtuse and produces a thicker stroke near the corner in **(b)** when compared to **(a)**.

quality of the smooth outlines that are discussed in the subsequent section. Note that the segment-end ribs for a vertex do not actually pass through the vertex, but since our prototypes are always rectangles, only the rib endpoints matter.

The folds generated by this method remain through the stroke-smoothing step described in the rest of this section, and are resolved in the layering method described in Section 6.2.

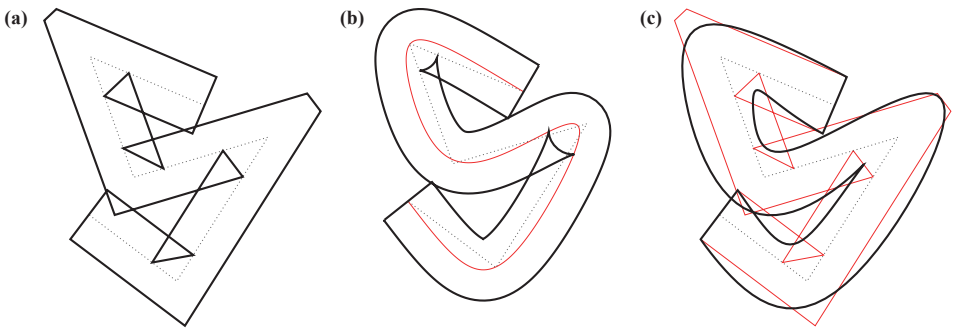


Figure 6.6: Variations of strokes for the same spine and width. **(a)** Polygonal stroke. **(b)** Curved stroke from the smoothed spine (in red). **(c)** Smoothed outline from the polygonal stroke (outline in red).

6.1.1 Smooth strokes

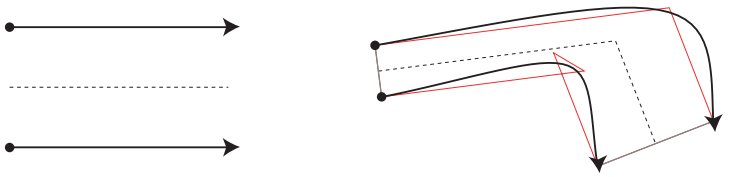
In addition to applying strokes to polygonal spines, we would like to smooth these to achieve certain graffiti styles. One approach would be to smooth the spine before applying the stroke, but we note that the result often looks rather mechanical and not hand-drawn (Figure 6.6b). Instead, we first deform a polygonal prototype along the original spine, and then treat the resulting polygonal outline as a motor plan that drives the generation of a smooth stroke using one or more MIC trajectories (Figure 6.6c). This is easily done by assigning one Gaussian to each polygonal stroke vertex.

Different types of strokes and stylisations can be produced by either varying the kinematics of the motion with the same techniques demonstrated in Chapter 5, or varying the shape of the prototype used to generate the stroke, which results in a variation of the motor plan. This allows for a large range of stylistic variations of a stroke. They resemble graffiti art visually and also mimic the process typically followed when constructing graffiti letters. Furthermore, spines usually consist of a small number of vertices, making them easy to author interactively or procedurally.

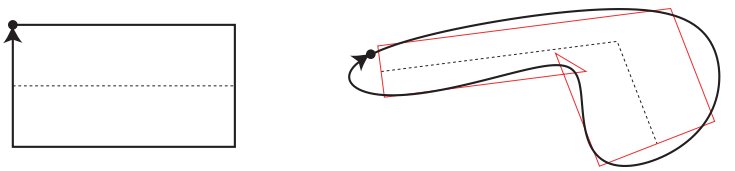
Smooth stroke types. We define three kinds of smooth strokes:

- *Squared end strokes* are defined with a prototype made by two parallel and similarly oriented lines. The stroke is then produced by tracing each side of the stroke with two separate motions, and then connecting the trajectory ends with two straight line segments.

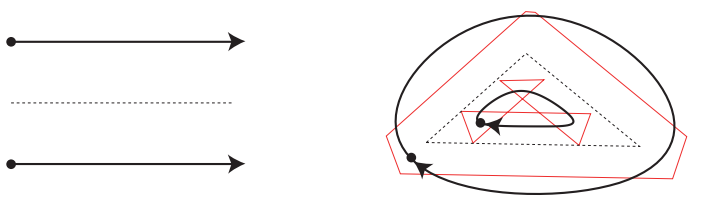
- *Rounded strokes* are defined by starting from a rectangular prototype. The stroke is then produced with a single periodic motion that follows all the vertices of the resulting skeletal stroke and returns to the beginning. The roundness at the stroke ends can be



parametrically controlled by adjusting the covariance matrices corresponding to the vertices at the ends of the stroke (Figure 6.7).



- *Closed strokes* are defined with a prototype made by two parallel lines and a closed polygonal spine. A closed stroke is then produced with two periodic motions that follow each side of the deformed prototype.



Some graffiti styles alternate smooth parts of a stroke with polygonal ones. To do so, we generate smooth trajectories for subsets of the envelope and connect these into a single stroke (Figure 6.8).

6.2 Apparent layering and overlaps

Graffiti often contains intricate overlaid and intertwined parts with non-global layering (Figure 6.2). These compositions can be evocative of a 3D projection, but rarely follow the rules of projective geometry, representing an abstraction or caricature of such rules. A systematic analysis of the geometry of the "pictorial space" (Koenderink, 2012) used in graffiti is left for future research. However, we can exploit the stroke-based structure to develop a 2D interface that allows self-overlaps (Section 6.2.2) and local layering (Section 6.2.3) which prove useful in applications.

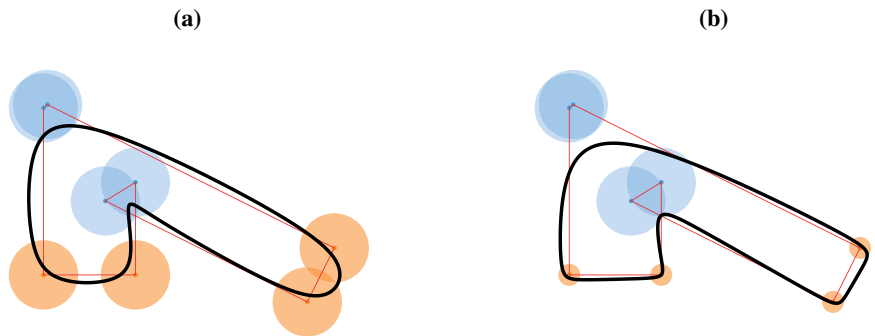


Figure 6.7: Rounded strokes. **(a)** The disks depict the covariance ellipses for the vertices of the stroke, the orange ones being for the stroke ends. **(b)** Decreasing the variance of these orange disks reduces the roundness at the stroke ends.

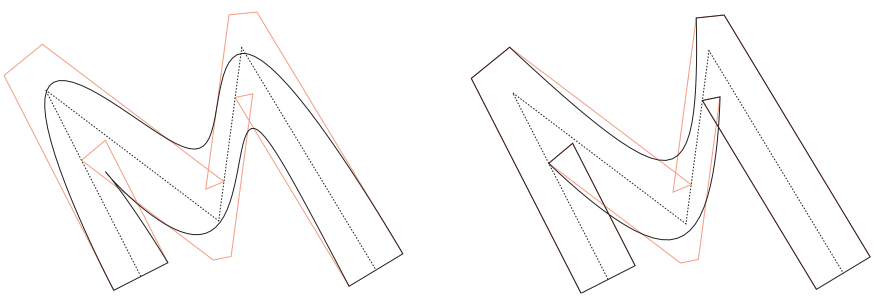


Figure 6.8: A smooth stroke with squared ends (left) and a piece-wise smooth version of it (right)

6.2.1 Partitions

Our layering and self-overlap rendering method relies upon subdividing strokes into a series of *partitions*. Each such partition corresponds to a connected portion of the spine representing a potential layer with an associated depth value. Not all depth assignments are possible; for example, in Figure 6.9, partition 1 cannot come between partitions 3 and 4 in depth order. Section 6.2.3 discusses how we prevent impossible assignments. Partitions are also assigned integer indices, in order, from the beginning of the spine to the end.

The skeletal stroke algorithm allows a straightforward mapping from points on the spine to corresponding points on the stroke outline. These portions of the outline form the outside edges of *partition shapes* (Figure 6.9). For outside corners with bevels, we assign the bevel edge to the outline of both adjacent partition shapes, and for inside corners with retrograde segments, we assign the loop area to both, as shown in Figure 6.9.a. This ensures that the partition shapes fully cover the area of the stroke, sometimes producing small regions where adjacent partition shapes overlap.

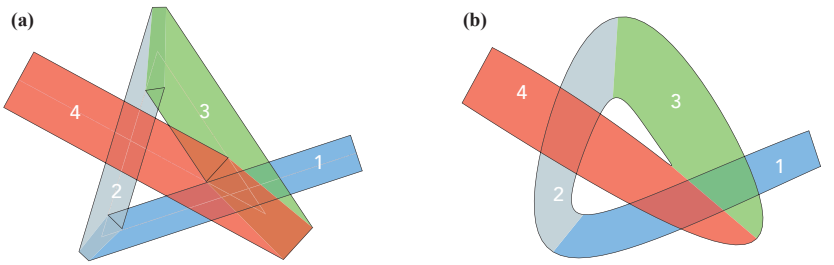


Figure 6.9: Partition shapes (color coded) for a **(a)** polygonal and **(b)** smoothly curving stroke.

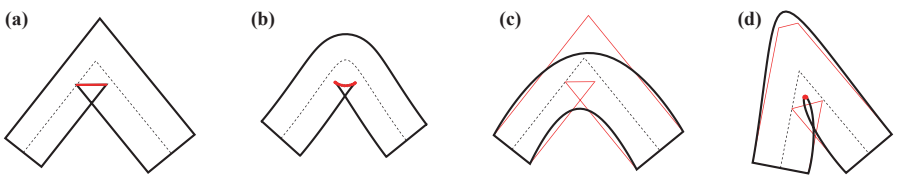


Figure 6.10: Different fold cases. **(a)** Corner in a polygonal stroke. **(b)** Curvature extrema in a stroke with a curved spine. **(c)** Flattened fold in a smooth stroke. **(d)** Fold in a smooth stroke. The retrograde portions are marked in red.

For polygonal strokes and spines the subdivision into partitions is trivial: each partition corresponds to a straight spine segment and the partition shapes are given by the outline segments mapped to each segment. For a polygonal spine with a smoothed stroke the partition shapes are given by mapping polygonal spine vertices to corresponding vertices in the smooth stroke outline. An initial estimate of these vertices is given by trajectory points corresponding to the passage time of each Gaussian. We then refine this assignment by identifying the closest curvature extrema to each pre-identified point. This method can be extended to other curve generation methods as long as a mapping is possible between the control points and the resulting curve. For an arbitrarily curved spine the partitioning is based on an estimate of curvature extrema and corners along the spine.

6.2.2 Fold culling

Deforming a prototype according to the proposed method often results in a shape that contains self-folds. Most traditional implementation of skeletal strokes consider this a problem and suggest methods to overcome these (Asente, 2010; Hsu and Lee, 1994; Lang and Alexa, 2015). In our application, we exploit this property to generate soft strokes (as discussed above) as well as to achieve stylised folding and overlap effects that are often seen in graffiti art, as well as in comics and other cartoons.

We identify folds with a procedure similar to the method proposed by Asente (2010) and find portions of the stroke outline that include retrograde motion. For a polygonal spine,

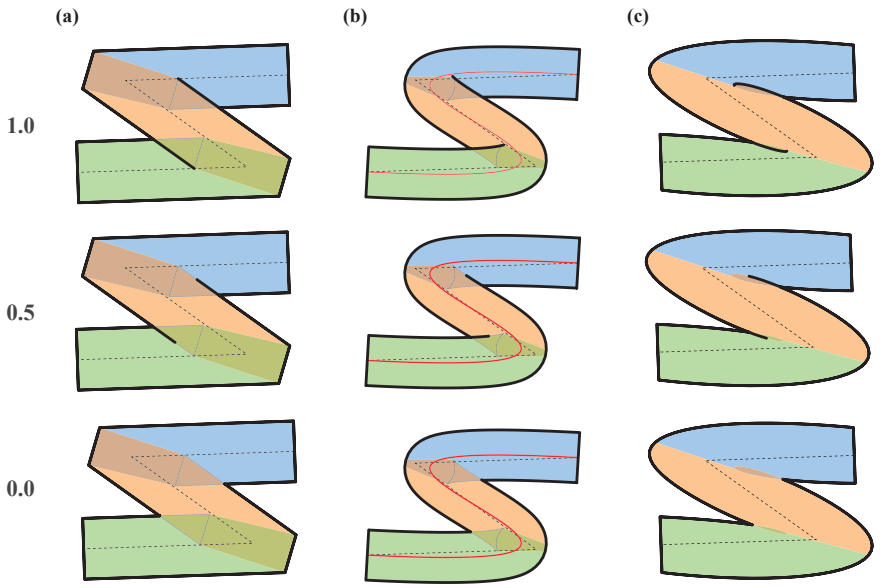


Figure 6.11: Stylised folds, showing the effect of the fold-rendering parameter.

these are trivially given by outline segments that are part of a concave portion of the outline and connect two vertices belonging to two different partitions (Figure 6.10.a). For a curved spine, these are given by the outline points that map to points of the spine with a radius of curvature less than the corresponding stroke half-width (Figure 6.10.b). For the case of a smoothed envelope, we first identify a series of potentially retrograde portions by finding the smoothed points that map to retrograde portions of the polygonal envelope. However, our smoothing technique is sufficiently flexible that it does not always maintain the retrograde segments, so there may not be a fold anymore (Figure 6.10.c). To determine whether there is a fold, we check if the midpoint of a potentially retrograde portion is contained in both adjacent partition shapes, in which case the portion is considered retrograde (Figure 6.10.d).

Once all retrograde portions have been identified, we traverse the outline on each side until we reach a common point of intersection. We then cull the retrograde portion of the side with lower depth. The remaining side is marked as partially visible according to a user configurable parameter $\in [0,1]$ that interpolates the visibility of the side relative to its length (Figure 6.11).

6.2.3 Layering and Planar Map

The partitioning scheme allows us to compute a layering of one or more strokes analogously to the method of Igarashi and Mitani (2010) for 3D shapes on a plane. We compute a *planar map* from the combination of all partition shapes, where each partition corresponds to a

layer. Each edge of the resulting planar map is assigned to an edge of a partition shape and the corresponding partition index. For each interior face of the planar map we then compute a *partition list* L_P that indicates which partitions overlap the region defined by the face. This can easily be done by choosing a point inside the face and testing which partition shapes contain the point. We then sort the partition list according to an ascending depth order, and iterate over each face edge. An edge is marked as *visible* if the partition it belongs to is the same as the higher one in the depth-sorted partition list.

Resolving impossible layer orders. The procedure above is efficient and can handle many types of complicated layering structures. At the same time, there can be combinations of partitions and depth values that have no consistent layering solutions, especially in the neighbourhood of spine vertices (Figure 6.9.a). To resolve these cases, we use a *list graph* structure (Igarashi and Mitani, 2010; McCann and Pollard, 2009), which has a vertex for each internal face of the planar map and an edge for each pair of faces that are adjacent and share a common partition.

Impossible overlaps can be detected by examining the connected components of the list graph and checking for inconsistencies in the layer ordering across the corresponding faces. For each connected component we compute a list of partitions assigned to it and sort it by increasing depth. By construction, two partitions with indices p_i and p_j are adjacent in the stroke if $|p_i - p_j| = 1$. A connected component of the list graph contains an impossible overlap if any adjacent pair in the list is not contiguous in the depth sorted list. If an impossible order is detected, we proceed in a manner similar to Igarashi and Mitani (2010) and compute the maximum area covered by each partition and consider all permutations that do not contain impossible orders. We then choose the permutation with the lowest number layer of swaps, weighted by the area of each layer.

Mixing strokes and arbitrary vector inputs. This layering method relies on a partitioning of the input given by our stroke representation. However, we can also combine strokes with arbitrary shapes (Figure 6.12.a) as long as each is treated as a single partition with a unique depth value, in which case the method operates as a vector counterpart of the one proposed by McCann and Pollard (2009) for bitmap inputs.

Unions. In addition to the depth ordering, we can also easily handle unions between one or more layers. To do so we define a set of union pairs $\{p_i, p_j\}$ between partitions, and cull an edge if any pair of partitions assigned to it correspond to a union. For example, we can add arrowheads to a stroke — an effect often seen in graffiti — by simply generating an arrowhead shape and then specifying a union between the arrow head and the partition corresponding to the end of a stroke (Figure 6.12.b). The same approach can be used to append arbitrary caps to the strokes with an effect similar to the one proposed by Jakubiak et al. (2006). A similar procedure should be possible also with other boolean operation, such as local differences or intersections, but this is left as an avenue of future development.

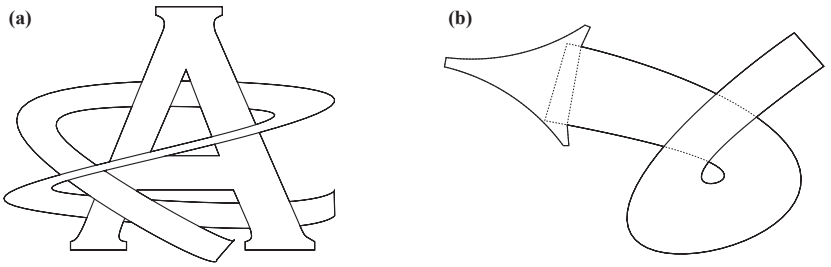


Figure 6.12: Additional layering effects. **(a)** A stroke is combined with the outline of a letter "A". The letter is assigned a single partition and depth value. **(b)** A union operation is used to add an arrow head to a stroke.

6.3 Results and Applications

The combination of the parametric stroke model and the proposed folding and layering methods lets us easily render intertwined strokes in a way that would be difficult to achieve with traditional vector graphics methods. The stroke representation can be constructed and edited with a simple interface and is well suited for the rapid generation of compositions and renderings that mimic the appearance of graffiti art.

Performance and interaction. The stroke generation and layering procedures can be used interactively and let a user quickly produce and explore variations of graffiti compositions. To test the performance of the method we generated patterns of increasing complexity, similarly the one shown in Figure 6.17. On a commodity laptop, we achieve frame rates suitable for interactive editing as long as the number of curve samples is fewer than 1000 (Figure 6.13). For example the letters in Figure 6.16 have about 400 points each and take less than 30 milliseconds for layering and rendering. The main bottleneck of the system is currently the curve generation method (Chapter 5), when the solution is computed with the least squares approach. For the case of squared end strokes, the solution can be computed iteratively, in which case the performance hit of smoothing is negligible compared with the layering procedure (Figure 6.13, left). When generating closed curves the least squares solution is necessary, and performance is mostly affected by the smoothing procedure (Figure 6.13, right). The pattern in Figure 6.17 is generated with a single closed stroke, has 7488 vertices and takes 10 seconds for curve generation and 0.2 seconds for layering. For interaction and preview purposes we can limit the number of curve samples, allowing for interactive editing of complex patterns like the one shown in Figure 6.1.c, which was produced interactively with our UI.

The interface to our method is simple: the user creates a stroke by clicking to define a sparse sequence of spine vertices. The user can then vary the shape of a stroke by adjusting stroke parameters such as the amount of smoothing. The width of a stroke can be adjusted globally with a set of sliders, or locally by dragging perpendicularly to a spine edge. The layering interface lets a user perform layer swaps in a manner similar to the one described by

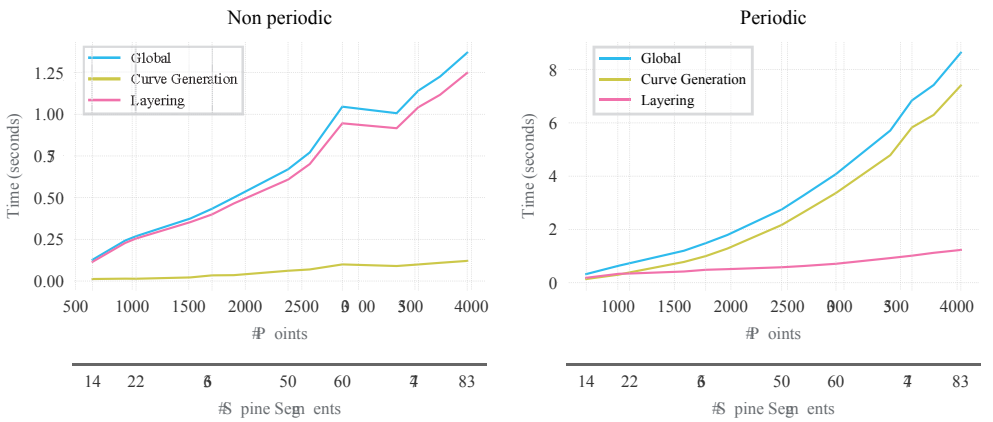


Figure 6.13: Performance of the method for increasing number of curve samples and spine segments. *Left:* non-periodic. *Right:* periodic.

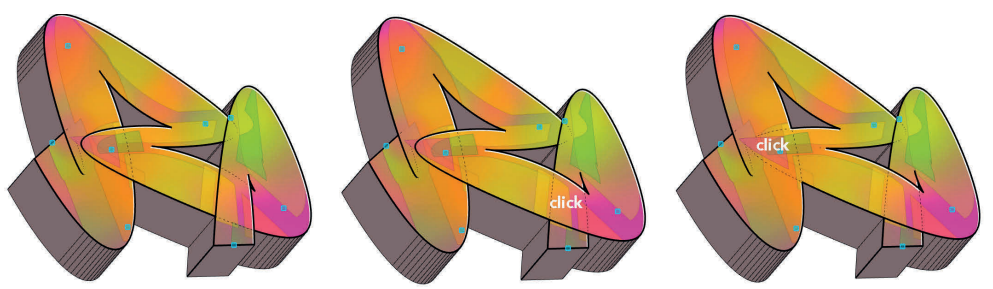


Figure 6.14: Layering interactions: with a single mouse click, the user can swap depth ordering (middle) or create unions (right).

Igarashi and Mitani (2010). Clicking on an overlap area brings the bottom-most partition to the top. Unions can also similarly be created by clicking on an overlap area with a different tool we provide. (Figure 6.14)

Fills and rendering effects. We generate colorful compositions by exploiting the faces of the planar map generated during the layering process. Randomly offsetting the faces and assigning each face a color from a user-specified palette gives results similar to those often seen in graffiti art (Figure 6.15.d). We can use the same palette to smoothly fill areas in ways that mimic the diffused use of spray paint. This can be simply done by using the union of all outlines to mask a raster fill. In the examples given here we generate the fill by randomly alpha-blending smooth gradient bitmaps over the interior of the outlines (Figure 6.15.c). To increase the realism of the rendering we can add highlights to parts of the outline that are approximately perpendicular to a given light direction. This, combined with an extrusion

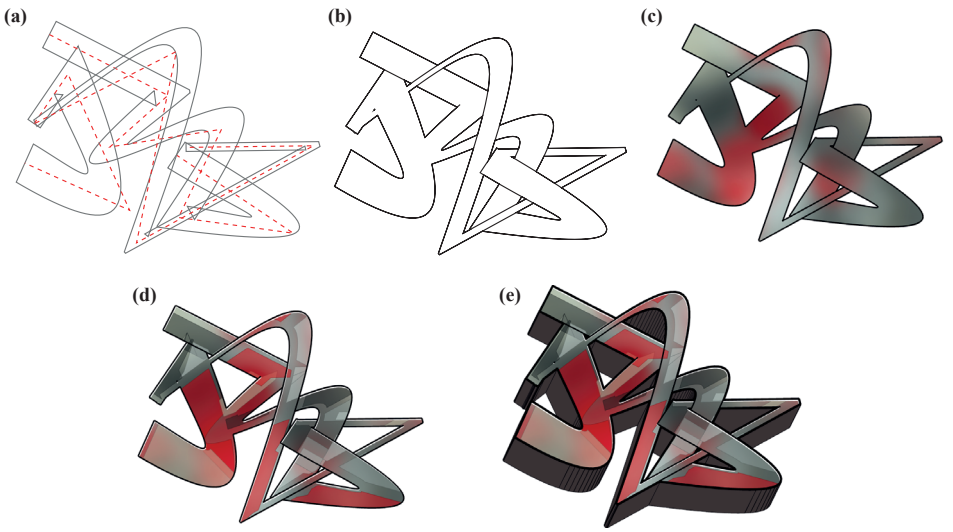


Figure 6.15: Interactive construction of a graffiti letter "R". **(a)** Stroke outlines and polygonal spine. **(b)** Layered outlines. **(c)** Fill-in gradients. **(d)** Geometric effects using planar map faces and highlights. **(e)** Extrusion.

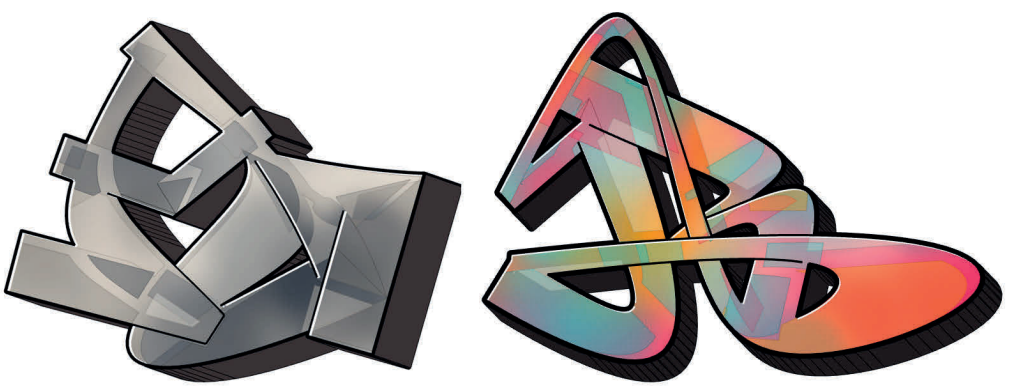


Figure 6.16: Graffiti letters (“A” and “R”) generated and rendered with our method.

effect captures a visual effect that is often seen in conventional instances of graffiti art (Figure 6.15.e). Figure 6.16 shows results that combine all these effects.

Extrusion. One effect that is often seen in graffiti art (Figure 6.2) is a simple oblique isometric extrusion of the composition as a whole. Our proposed method to construct these digitally proceeds in 7 steps as follows:

1. Rotate the entire composition so that the extrusion can be done directly downward, in the negative y direction.
2. Create a planar map from the rotated composition and extract the edges.
3. Split each edge at corners and at extrema in the x direction. The result is a set of straight and curved segments that intersect each other only at their endpoints. Each segment has x coordinates that monotonically increase or decrease.
4. Perform a topological sort of the segments, with the ordering function being that segment s_1 is greater than segment s_2 if some point on s_1 and some point on s_2 have the same x coordinate, different y coordinates, and the point on s_1 has a larger y coordinate. This formalizes the idea that s_1 is greater than s_2 if s_1 is higher than s_2 in the y direction.
5. Construct a total order from the partial order produced by the topological sort.
6. For each segment, construct an extrusion face by offsetting the segment vertically by the extrusion depth and connecting the ends of the original and offset segments. Stack these faces with the last – the one from the edge with the smallest y coordinate – on the top of the stacking order.
7. Place the extrusion faces below the rotated composition, and rotate everything back to the original orientation.

The extrusion faces can then be stroked and filled as desired. Figures 6.1, 6.14, 6.15 and 6.16 show examples produced by our system. The extrusion can be modified by choosing areas to subdivide more finely, creating an effect similar to that in Figure 6.2, top.

Figure 6.14 also includes a thicker outline around the union of all the strokes, another common effect seen in street art (e.g. Figure 6.2, bottom).

Generative applications: Weaving patterns. Specifying a stroke with a sparse sequence of control points is a simple user interaction procedure. The same sparse representation is also convenient in a procedural modelling applications, in which the system can operate at a high level by specifying the sparse sequences of control vertices. Then, various stylisations of the output can be explored parametrically. For example, a simple procedure can generate stylized knots or weaving patterns. We first generate a 2D lattice (Figure 6.17a) and compute an

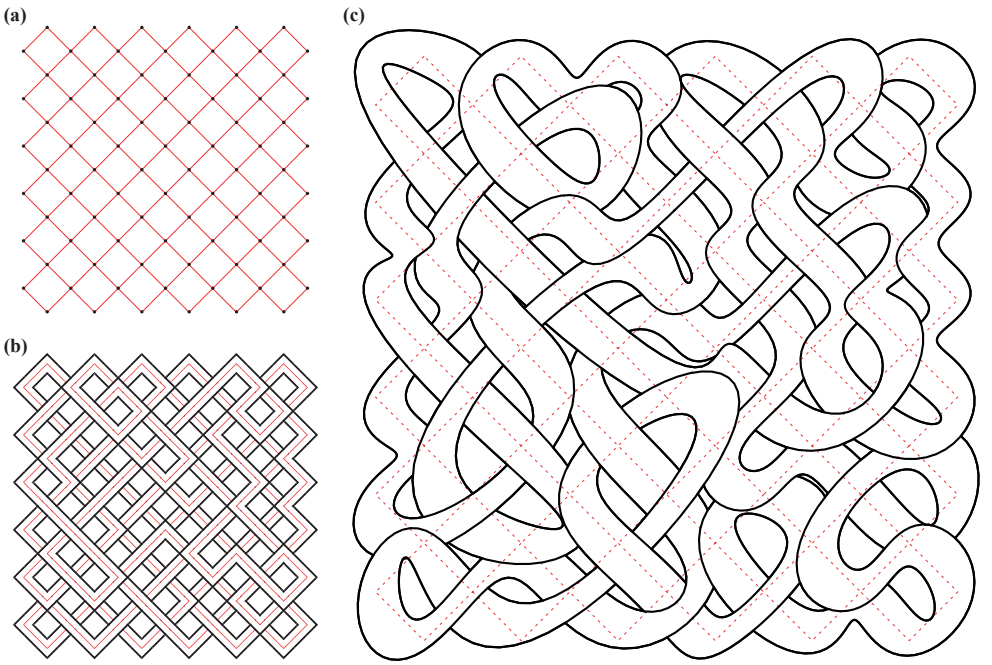


Figure 6.17: Weaving pattern generated by constructing an Eulerian cycle **(b)** over a planar graph **(a)** and then using the resulting path to generate one looping stroke **(c)**.

Eulerian path or cycle along the lattice (Figure 6.17b). We then construct a single stroke along the path, making sure that the depth values of crossings are interleaved (Figure 6.17c). The results are evocative of more abstract forms of graffiti art and sketchy renditions of weaving patterns.

Machine drawings. The output of our method is suitable for being realized with a drawing robot or plotter. Once the primary printing tool in the early days of computing, plotters have today regained popularity as a creative tool for computer graphics because of their affordability and their ability to create vector drawings using a variety of physical drawing media.

The output of our method is suitable for constructing tool paths for such machines. Furthermore, since we maintain the path ordering defined at the stroke level, the motions of the machine are visually consistent and often evoke the sequence of movements that would be followed by a human when producing a drawing (Figure 6.18). This same property could be used to generate stroke animations from the output of our system.

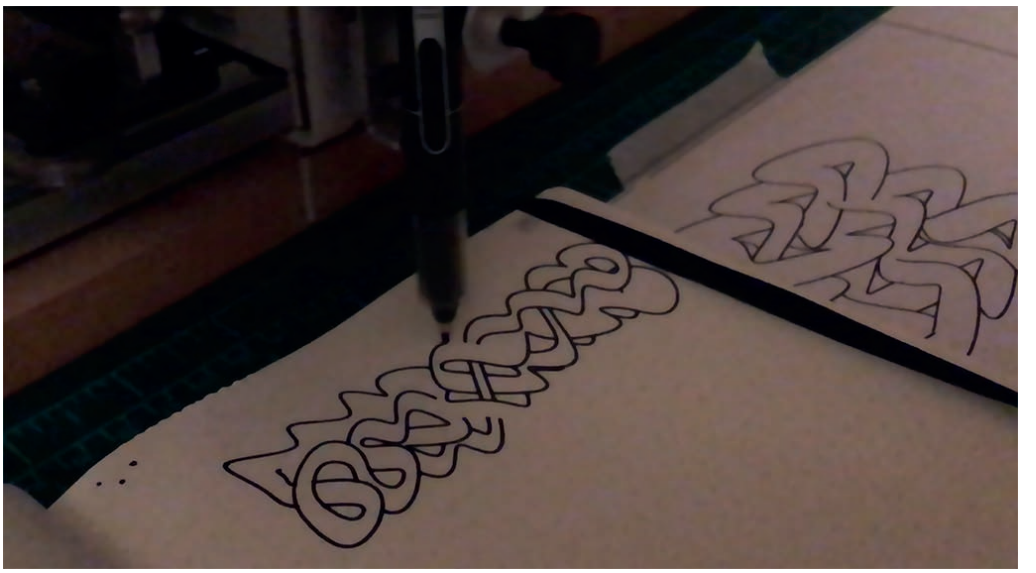


Figure 6.18: A pen plotter drawing a weaving pattern generated by our system running on a notebook.

6.4 Conclusion

We presented a system and interface that permit the generation of convincing synthetic graffiti with simple, flexible stroke representation. Our method generates strokes with self-overlap effects that are typical of this art form. However, our current approach builds on the assumption of a rectangular skeletal stroke prototype. Extending this approach to arbitrary vector inputs is an interesting extension for future studies, but doing so is not trivial. With complex prototypes (Figure 6.19, left) folded areas can occur within the stroke, and not just along the boundary. One possible way to handle these cases could be to estimate the spine through the computation of symmetry axes (Blum and Nagel, 1978). Symmetry axes also have the potential to extend our layering procedure to arbitrary shapes. While our current approach relies on the partitioning of the input given by the skeletal stroke spine, the same partitioning could be computed automatically from the skeleton of the input.

In our description of the layering procedure, we have focused on the generation of outlines. In future developments it would be desirable to handle the layering of fill patterns and gradients defined along a stroke as well. This can be achieved by exploiting the planar map and partition list generated during our method. The output of our method is a coherent sequence of outlines, that can be used for machining applications or to create animations of the reconstruction of the outlines. However, because of the planar map subdivision, the outlines do not reflect smooth movement kinematics. Producing an entirely smooth reproduction of the outlines is an interesting avenue of future research and would result in a method that reflects more closely the process used in live constructions of graffiti letters and would potentially produce more organic stylisations of letterforms. At the same time, our current

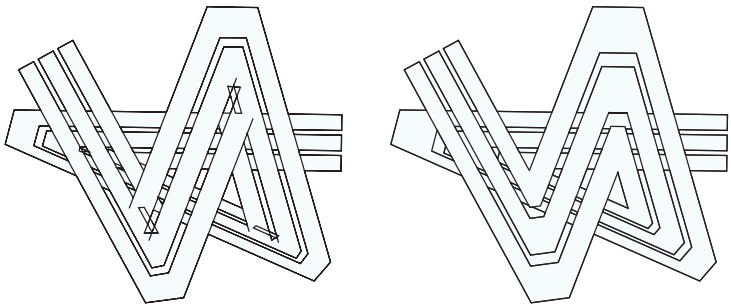


Figure 6.19: Layering of strokes with a more complex stroke prototype. On the left, our self-overlap procedure fails because folds occur within the stroke, and not just along the boundary. On the right, we shrink the ribs along the bisector, as in Hsu et al. (Hsu and Lee, 1994) and as discussed in Section 6.1, resulting in a visually consistent output.

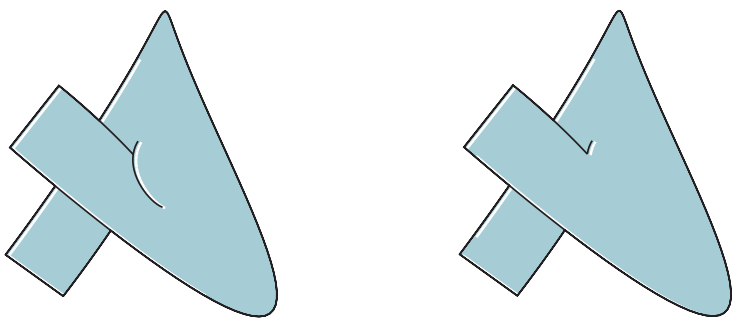


Figure 6.20: Overlaps with self-folds. We currently support this effect only through user interaction. The effect is prohibited by the impossible layer order resolution procedure (right).

approach allows to reproduce a number of stylistic features that are often seen in graffiti art. For example the stylised letter forms seen in Figure 1.5, in the introduction (Chapter 1), have been generated with this method and only required a few clicks and adjustments to create; this is achieved with a procedure that is compatible with the one found in standard stroking methods in popular software packages such as Illustrator or InkScape.

The layering procedure combined with fold culling can automatically and rapidly handle many different configurations of one or more strokes with self overlaps. However, our current implementation of impossible layer resolution (Igarashi and Mitani, 2010) does not permit certain configurations that visually make sense. As an example, it may be desirable to render a partition of a stroke that passes below a fold produced by the adjacent partition (Figure 6.20, left), but this effect is discarded by the resolution process (Figure 6.20, right). In order to handle these cases, we currently allow the user to disable the resolution step and create this effect by performing layer swaps with a few clicks in the regions of interest. In

future iterations of this work we plan to handle these cases automatically.

The interface developed in this chapter completes a system in which a user is able to define a variety of different graffiti styles with a few clicks, and fine-tune the results with parametric variations of the stroke primitives. In the next part of the thesis, we will recover these primitives from existing geometry. As we shall see, doing so results in a flexible procedural generation and stylisation system, which produces outputs that can be edited with exactly the same procedures that have been discussed in the preceding part.

Figure 6.21: Weaving pattern deformed along a spine and then stylised.

Part II

Part II - Graffitization: Recovering graffiti primitives from shape

Chapter 7

Curvilinear Shape Features

> "Since Picasso has some of the most advanced use of lines in the history of art, this inevitably means that his central tool is the use of curvature extrema"
>
> Michael Leyton,
> The structure of paintings
> (Leyton, 2006)

This chapter introduces the definition of Curvilinear Shape Features (or CSFs). It is the fruit of brainstorming and many discussions held between myself and Prof. Frederic Fol Leymarie since the onset of my PhD studies, as we sought better ways to characterise the shape of curve traces and contours. This chapter refines and expands an earlier definition and implementation of CSFs, which appears in one conference publication (Berio et al., 2018b) and one book chapter (Berio et al., 2020b).

7.1 Introduction

The methods described in the next few chapters depend on the recovery of a plausible set of stroke primitives that reconstruct an input traces and outlines. Doing so extends the previously described interactive editing, variation, and stylisation methods to arbitrary vector inputs, laying the foundation for a flexible graffiti and calligraphy stylisation and generation framework. The types of the input can vary from open digitised traces of tags, handwriting, drawing, or curves defined in a vector drawing package, to the closed traces of glyphs and other 2D object outlines.

We have seen in Chapter 3, how extrema of curvature (i.e. with associated contour segments having a role of support) are the most salient loci along piecewise smooth contours

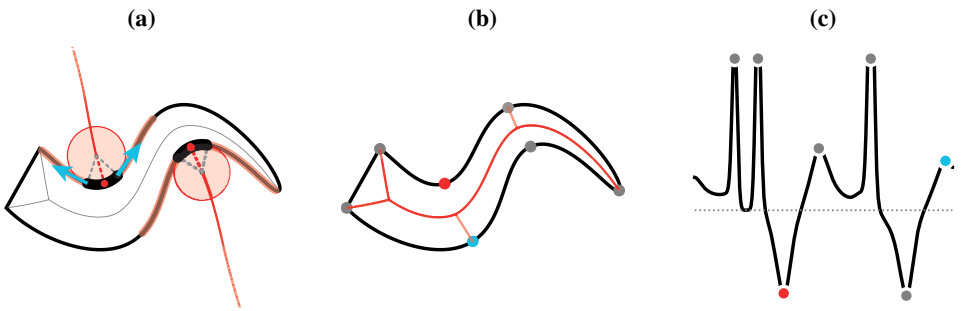


Figure 7.1: (a) Interior SA and two concave *curvilinear shape features (CSFs)* for a closed contour. Each CSF consists of an extremum (red circle), a local symmetry axis terminating at the extremum (red axis), a contact region where curvature is approximately constant (thick black segment), and two *support* segments (thick red segments). The blue arrows show tangents computed at the beginning of the support segments of the left CSF. **(b)** All CSF extrema (circles along contour) and the corresponding *curvilinear augmented symmetry axis* (CASA), which has two new branches that terminate at convex features. The additional branches allow to easily identify morphological features, such as the bend characterised by the red and the blue extrema that are "perceptually close" (Singh, 2015). This would be more challenging with a classic contour-only based approximation of curvature **(c)**, where the two extrema are far apart in a 1D traversal of the contour.

(Attneave, 1954; Feldman and Singh, 2005; De Winter and Wagemans, 2008b) and play an important role in the perceptual decomposition of 2D objects (Richards and Hoffman, 1985; De Winter and Wagemans, 2006) or signed traces (Brault and Plamondon, 1993a,b) into parts. It follows that an informative analysis of curvature is of key importance for the stroke recovery task ahead of us. In Chapter 8, we seek to recover kinematics from the geometry of a trace in terms of a motor plan and a set of associated $\Sigma\Lambda$ parameters. Given the stereotypical inverse relation between movement speed and curvature in human hand movements (Viviani and Schneider, 1991; Plamondon and Guerfali, 1998a), curvature extrema and the corresponding radii of curvature are useful features for determining the number of generating sub-movements and $\Sigma\Lambda$ parameters for a trace. In Chapter 10, we seek to recover a set of strokes that reconstruct the outline of a glyph. Again, curvature extrema and their associated outline regions will prove useful to infer perceptually meaningful parts, to merge these parts into potentially overlapping strokes and to characterise stroke morphology.

To guide these reconstruction tasks, we seek a feature representation that robustly identifies salient extrema while providing a precise estimate of the associated center and radius of curvature, as well as an associated region of influence. To this end, we propose an alternative to traditional methods based on curvature retrieval, which relies on the geometric analysis of *local symmetries* rather than a filtering approach based on calculus (along a curve). We

exploit the duality (Leyton, 1987) between the two representations of (2D) contour curvature and Symmetry Axis Transform (SAT), which allows us to identify *significant* curvature extrema and discontinuous breaks along an open or closed contour in terms of a set of features we call *curvilinear shape features* (CSFs). Figure 7.1.a displays two such features along with the interior medial axis of a simple object. Each CSF identifies an absolute maximum of curvature along a trace (red dots, $M+$ or $m-$) together with a circular arc segment surrounding the extremum where curvature is approximately constant (thick black segment). In addition, each CSF is also associated with a local symmetry axis (red) terminating at the extremum and a pair of curvilinear support segments (light red) — trace segments on each side of an identified extremum.

The support segments localise the influence of a CSF and facilitate the analysis of additional contour features such as tangents near extrema (Figure 7.1.a, blue arrows) or remaining curvature features such CSFs for absolute minima of curvature ($m+, M-$), as well as permitting the robust localisation of inflection points. This feature set enables a full reconstruction of the curvature function in terms of segments with monotonically varying curvature, bearing similarities to the representations proposed by Leymarie and Levine (1989) and Baran et al. (2010).

For closed contours, we also use CSFs to compute a *curvilinear augmented symmetry axis* (CASA), an augmented version of Blum's SA (Blum and Nagel, 1978) that, in contrast to the conventional formulation (Belyaev and Yoshizawa, 2001), is guaranteed to have branches terminating at all absolute maxima of curvature. This results in a mixed contour+region representation that allows to relate features that are nearby on the shape but distant when considering a 1D traversal of the contour. As noted by Singh (2015), these relations are difficult to distinguish with an analysis of the curvature function alone. As an example, consider the two curvature extrema emphasised with red and blue dots in Figure 7.1.b, alongside with the remaining curvature extrema (gray dots) and the CASA of the object interior (red). The two emphasised extrema identify a region where the object "bends" and are related by one CASA branch. The branch terminates at the convex extremum, while the other extremum is located on the opposite side of the branch. In contrast, the same two features are seemingly unrelated when only taking into consideration the curvature function in Figure 7.1.c. We sill see in Chapter 10 how these semantic relations between CASA and CSFs are easy to identify computationally and how this can be integrated into an automatic method to recover strokes from glyph outlines.

7.1.1 Masking Problem

In order to identify CSFs, let us first recall the result proved by Leyton, which links the symmetry axes of an object having a smooth bounding contour to its curvature extrema (Leyton, 1987):

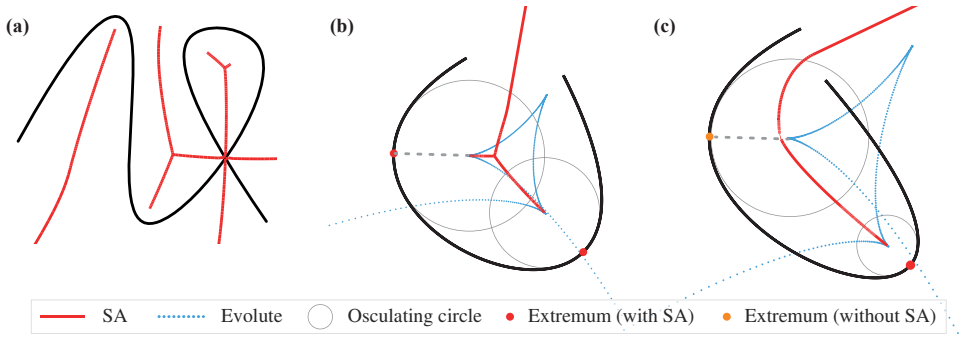


Figure 7.2: Issues with the SAT for the identification of curvature extrema. **(a)** the *SA* (red) of a self intersecting trace. **(b,c)** Evolutes (blue) and the masking effect as demonstrated by Belyaev and Yoshizawa (2001). In **(b)**, the segment (dashed gray) connecting the right curvature extremum (red circle) to its center of curvature does not intersect the SA, and the corresponding osculating circle (light gray) does not intersect the outline. Thus, the SA identifies the extremum with a branch terminating at its centre of curvature. In **(c)**, the segment intersects the SA and the osculating circle intersects the trace. Consequently, the SA branch that would identify the left curvature extremum (orange circle) is not present on the SAT.

Any segment of a smooth planar curve, bounded by two consecutive curvature extrema of the same type, has a unique symmetry axis, and the axis terminates at the curvature extremum of the opposite type.

This result holds also for cusps and corners (Hayes and Leyton, 1989), suggesting that given a symmetry axis (*SA*), it is possible to identify and locate a curvature extrema near one axis end. However, doing so is difficult when computing the SAT to a trace as a whole, and indeed the analysis given by Leyton is local and takes into considerations *codons* (Richards and Hoffman, 1985), i.e. curve segments defined by consecutive triplets of curvature extrema. This approach assumes that knowledge about the curvature behavior of an outline is a priori available, which is rarely the case in practice.

Furthermore, the global SAT structure is linked to the overall geometric and topological configuration of the trace. As an example, a handwriting trace is likely to have self intersections, which will produce symmetry axes that terminate at intersections instead of curvature extrema (Figure 7.2.a). In addition, the global SAT is not guaranteed to identify all perceptually significant curvature extrema, as part of a contour can forbid or *mask* the existence of a *SA* that would otherwise end at a curvature break or corner, or end at the center of curvature of the circular arc associated to a curvature extremum.

Belyaev and Yoshizawa (2001) prove this masking effect for smooth closed curves in terms of the *evolute*, i.e. the trace of the center of curvature of the osculating disks (Figure 7.2.b in blue). The evolute always has cusps corresponding with curve vertices (i.e. curvature

extrema). Belyaev and Yoshizawa (2001) show that an evolute cusp corresponds with a *SA* branch only when the segment going from the cusp to the associated curvature extrema does not intersect the *SA* (dashed grey segment in Figure 7.2.b). When the radius of curvature of the extremum is sufficiently large, this segment intersects the SAT and the *SA* branch that would otherwise identify the extremum "disappears" (Figure 7.2.c).[1] This can be interpreted in terms of the "maximality" requirement imposed by the SAT definition, so any curvature extrema the osculating circle of which intersects the trace will not not be part of the of SAT and thus it will not result in a terminal *SA* branch.

One possible solution to avoid the masking problem is to identify features with the full Symmetry Set (SS) (Giblin, 2000; Giblin and Kimia, 2003), which, for the case of smooth and regular curves, has terminal disks centered at evolute cusps; however, the SS creates much more complex diagrams where a large part of the structure is (alike the SAT) linked to global symmetries and other geometric and topological features of the trace input. Also, the retrieval of the SS is much more involved, and only a small amount of attention has been devoted to its computation (Kuijper et al., 2006); and there is no known use in practice.

7.1.2 Solution: Recursive CSF Computation

The proposed practical solution is simple. First estimate an *initial* CSF set from the SAT computed globally, or, for the case of traces with self intersections for the SAT computed locally for a set of non-intersecting trace segments. Then compute an *additional* CSF set by recursively visiting the support segments of previously identified CSFs and computing a *local SAT* for each such segment. This procedure terminates when no new CSFs can be found. The proposed solution effectively avoids limitations of both "extremes" represented by the SAT (masking, and branches terminating at self-intersections) and the SS (complexity, difficulty of implementation) while providing an easy to manage and complete descriptor of curvature extrema with a representation similar to the one proposed by Leyton (1987).

In the following sections, we first introduce a more precise definition and implementation details for the SAT (Section 7.2), CSFs (Section 7.3) and CASA (Section 7.3.4). While the basic definition of CSFs covers only absolute maxima of curvature, in Section 7.4 we show how this definition can be extended to include absolute minima in order to cover all curvature extrema types ($M+, m-, M-, m+$). We finally show in Section 7.5 how to reconstruct the remaining contour segments with Euler Spirals, which identifies inflections and results in a piecewise linear approximation of the curvature function.

[1] In such a configuration there is no space left for the formation of that *SA* branch; e.g. this is easily understood when thinking of the equivalent grassfire propagation process to generate the SAT: there is no grass left to burn.

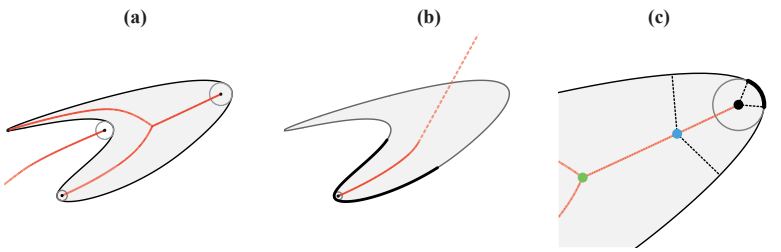


Figure 7.3: Global and local SAT. **(a)** Traditional global SAT of a closed contour. **(b)** Local SAT for a given CSF (dark thick contour segment centered at a convexity, bottom left). **(c)** Detail of the right side of the figure and SAT in **(a)**. (i) The green dot on the left identifies a terminal branch starting at a fork and ending in a terminal disk (black dot). (ii) The corresponding contact region (thick black arc) (iii) Some ribs (dashed lines) are shown, here emanating from a regular *SA* point (blue dot) and a terminal point (black dot).

7.2 Symmetry axis transform

The symmetry axis transform (SAT) of a set of input traces, is the set of all *maximal disks* that are tangent [2] to the trace in at least two distinct points (including limit points at corners), without intersecting the trace set itself. The *symmetry axis*, *SA*, is the set $\{\boldsymbol{\rho}\}$ of all *symmetric points*, the centers of the SAT disks. *SA* points also have an associated radius function $r(\boldsymbol{\rho})$ (from maximal disks). The *SA* is typically organised as a (directed) graph in 2D and as a hypergraph in higher dimensions (Leymarie and Kimia, 2007), with flow directions provided by $r(\boldsymbol{\rho})$. The *global SAT* (resp. global *SA*), is the SAT computed simultaneously for a set of one or more traces (Figure 7.3.a). The *local SAT* (resp. local *SA*), is the SAT computed for a connected segment along a given trace (Figure 7.3.b).

Symmetric points part of the *SA* can be categorized into three types depending on their *degree*, which is the number of nearest distinct trace points (Giblin and Kimia, 2003).

1. A *terminal* point has degree one and coincides with a corner or the center of a curvature extremum along the outline.

2. A *normal* (a.k.a. regular) point has degree two.

3. A *fork* point has degree three or more, and coincides with an *SA* locus which branches in three or more paths.

[2] Because we are dealing with traces that can be either open or closed, the definition of tangency is relaxed to include endpoints, cusps and corners. To do so, we can adopt Blum's definition of "pannormals" (Blum, 1973) and use these to measure the shortest distance to the trace set. Pannormals do correspond to normals for smooth trace segment, but generalise to "radials" that cover successive directions where gaps may occur for adjacent normals where the later are not defined, for example at the end points of a curve, at isolated sample points, or at sharp corners covering a conical range of normal directions.

Normal points can be visualized with *ribs* connecting them to the two nearest outline points. A *branch* is a series of connected normal points that ends in either a fork or a terminal point. Certain branches that are not bounded by a closed trace extend to infinity. A *terminal branch* is a branch that ends at a terminal point. A *terminal disk* is the *SA* disk centered at a terminal point. When a trace segment is a circular arc, the disk touches the curve over a finite *contact region* that coincides with the arc. When a terminal branch ends in a corner, its terminal disk shrinks to a point and so does the disk's contact region.

Object outlines. When computing the SAT for the contours of an object outline, we distinguish between the *interior* and the *exterior* symmetry axes, respectively denoted as SA^I and SA^E. The SA^I lies entirely within the object's figure, and its terminal disks coincide with positive curvature extrema or convex corners along the outline. A loop in SA^I is indicative of (surrounding) a hole in the object. The SA^E lies entirely in the object's background, and its terminal disks coincide with negative curvature extrema or concave corners along the outline. A convex outline produces no SA^E. The SA^E of non-convex objects typically has some branches extending to infinity.[3]

7.2.1 Discrete implementation

In the remainder of the thesis, we will use the term *trace* to refer to either open or closed sampled segments. However, for closed segments bounding a solid 2D object, we will also use the term "contour". Contours are assumed to be simple (no intersections or loops such as found in signatures or calligraphy) and consistently oriented according to the conventions described in Chapter 2.

The input data for our method consists of a set of one or more traces, each denoted as $\boldsymbol{z}(s)$ and parameterised by arc length s, each being representative of open or closed segments. To treat continuous, piecewise-continuous and discrete inputs with the same framework, we uniformly sample each trace giving a sequence of discrete points at an approximately equal distance Δs from each other. In the presence of corners or highly curved portions of an outline, other more sophisticated (adaptive) sampling strategies are possible, such as the classic iterative application of the de Casteljau algorithm for Bézier curves or other curvature-based techniques (de Figueiredo, 1995); but we have found that in practice a uniform sampling with a sufficiently small Δs works well for our use case.

In order to treat differently scaled inputs with similar thresholds, we first scale the traces so that the height of their bounding box h_{ext} is equal to a user defined amount. We use $h_{ext} = 150$ and $\Delta s = 1$ in the accompanying examples. The choice of the height to compute the scaling factor is motivated by the assumption that our input mostly consists of characters or text strings that are written horizontally. In the presence of noise, we observe that it is

[3] In practice such branches to infinity are usually cut-off at a maximum distance or for some pre-defined bounding box.

convenient to slightly pre-smooth each trace by a small amount that does not degrade its overall (perceived) shape.

7.2.2 Voronoi approximation

Many algorithms for computing the SAT exist, and the CSF identification procedure that follows can easily be adapted to any such method that operates on either curves or polylines. We choose to rely on a Voronoi-based approximation as described by Ogniewicz and Ilg (1992) because it is efficient, robust to quantisation noise, and well established.[4] The method supports a family of regularisation methods that discard superfluous edges based on a "potential residual" measure $s_{\min}$, the shortest geodesic length along the trace connecting any two symmetric points $\boldsymbol{z}(s_i)$ and $\boldsymbol{z}(s_j)$. When $\boldsymbol{z}(s_i)$ and $\boldsymbol{z}(s_j)$ are part of different traces, $s_{\min}$ is set to an arbitrarily large value. The examples given in this thesis use the "chord residual" regularization variant, where a Voronoi edge is discarded if $s_{\min} - \|\boldsymbol{z}(s_i) - \boldsymbol{z}(s_j)\|$ less than a user defined threshold. This pruning process effectively corresponds with removing Delaunay triangles that do not greatly contribute to the shape boundary. An example of the use of this approach is given in Figure 7.4 for a 2D letterform "a".

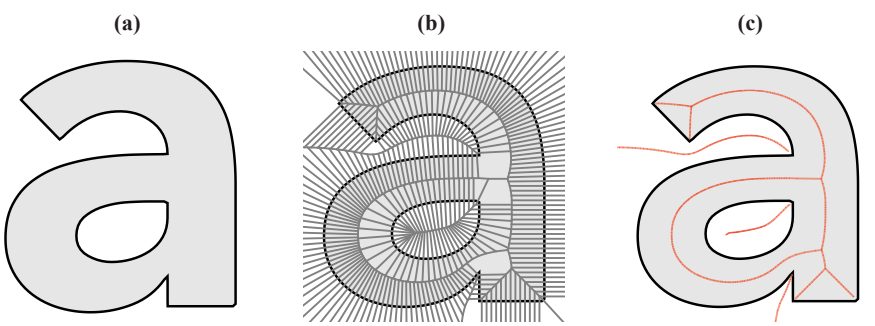


Figure 7.4: Voronoi skeleton (in (b)) to the SAT (in (c)) using the method of Ogniewicz and Ilg (1992).

The *SA* approximation is given by the edges remaining after regularization, where terminal points and forks coincide with Voronoi vertices of degree 1 and degree 3 or more, while normal vertices coincide with the midpoints of kept Voronoi edges. The disk radii for normal vertices are given by the Voronoi edge distance to any of its symmetric points, while the disk radii for forks and terminals are given by the circumcircle of the corresponding Delaunay triangle. The contact region of the approximated terminal disk is the shortest trace segment connecting its nearest points $\boldsymbol{z}_i, \boldsymbol{z}_j$, which corresponds to a circular arc that approximates the trace to within a small tolerance.

[4] I implemented the approach in Python using the QHull package (Barber et al., 1996) for robustness and accuracy.

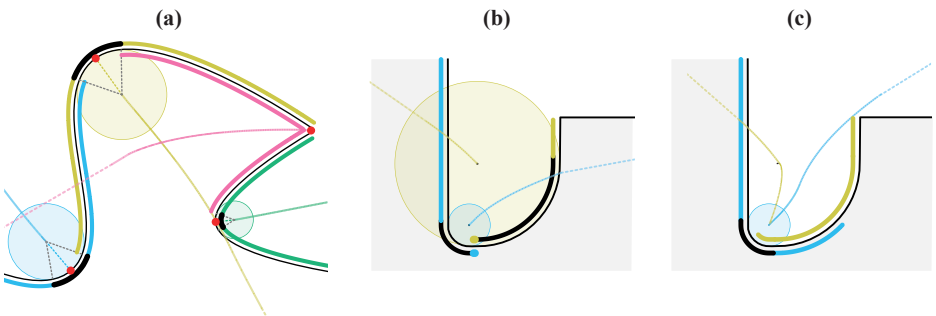


Figure 7.5: (a) Four successive Curvilinear Shape Features (CSFs): Each CSF is defined by (i) a local symmetry axis, extended to reach the outline, (ii) a terminal disk, (iii) its contact region (an arc or point), (iv) a representative curvature extremum, (v) a pair of supporting outline segments, each shared by the adjacent CSF. The third CSF (from the left) is a corner and thus has a contact region reduced to a point. The local symmetry axes can intersect and overlap, unlike medial axes. **(b)** A configuration, such as the one that might occur in a serifed letter E, where one CSF with a small radius (blue) is adjacent to one with a larger radius (yellow). This results in a very short support segment shared by the two features. **(c)** To compute CSF saliency, we extend the support segments to the adjacent extrema, which here helps capture the concave area around the blue CSF. We do not use the extended segments when identifying CSFs; doing so here would result in local symmetry axes that miss the larger yellow CSF.

7.3 Computing Curvilinear Shape Features (CSFs)

Curvilinear shape features are trace segments that identify curvature extrema, including sharp angles at corners.

Definition 7.3.1 (Curvilinear Shape Feature (CSF))**.** A CSF has five elements:

- SA_i, a *local* symmetry axis segment extended to reach the trace;
- CC_i, a terminal disk centered at the tip of SA_i before it is extended;
- $\overrightarrow{CC_i}$, the associated contact region (an arc, or a point for corners);
- $\hat{\boldsymbol{z}}_i$, the associated extremum of curvature, which we take as the mid-point of the contact region $\overrightarrow{CC_i}$;
- $\boldsymbol{z}_i^{lhs}(s)$ and $\boldsymbol{z}_i^{rhs}(s)$, a pair of supporting trace segments on each side of $\overrightarrow{CC_i}$, representing the CSF's region of influence.

The index i indicates one of N_{CSF} computed CSF: $1 \le i \le N_{CSF}$.

Each support segment of a CSF is the curve segment extending from one end of the CSF's contact region to the beginning of the contact region of the adjacent CSF if present, or to the adjacent endpoint otherwise (Figure 7.5a). Adjacent CSFs always share one support segment.

The local symmetry axis SA_i of a CSF is given by the SAT of the trace segment spanned by the CSF's contact region and its two support segments. Because this trace segment is open, one end of the axis extends to infinity and the other begins at the center of the terminal disk CC_i. We extend the axis with a straight segment connecting the disk center to the extremum $\hat{\boldsymbol{z}}_i$. This extension of a terminal branch results in an axis that is similar to the ones produced by the "Process Inferring Symmetry Axis" (PISA) as proposed by Leyton (1988), but avoids numerical precision issues when dealing with discrete traces. A given arc, $\overrightarrow{CC}_i$, may vanish in size when coinciding with a break of curvature or sharp corner tip, becoming identically $\hat{\boldsymbol{z}}_i$. We also emphasise that this definition of a CSF is more general than the older concept of a (contour-based) "codon": a triplet of curvature extrema (concave, convex, concave) (Richards and Hoffman, 1985).

7.3.1 CSF Computation

As previously mentioned, the CSF set consists of the union of an initial CSF set with an additional CSF set computed with a recursive estimation of local symmetry axes. When the input consists of multiple traces, the CSF set consists of the union of the CSFs identified for each trace separately.

Initial CSF set: With closed contours, the initial CSF set is trivially computed from the terminal disks of a contour's global SAT. When a trace contains self-intersections, a preprocessing step is used that splits the trace into a set of non self-intersecting segments. We currently implement this with a brute-force method that traverses the trace starting from one end-point and adds points to one segment until it intersects itself or the other end-point is reached. If a self-intersection point is found, a new segment is started from that point on. Then, the initial CSF set is given by the union of the CSFs sets produced by the local SATs computed for each so-identified segment.

Additional CSF set: The initial CSF set consists of a number of CSFs connected by support segments. As previously discussed this initial set is incomplete, since the SAT can miss important features, depending on the local configuration of a trace (Belyaev and Yoshizawa, 2001). To identify missing CSFs, we compute local SATs for each previously identified support segment, and consider the local CSFs that would be produced by its terminal branches. For each support segment, we first discard any local CSF with a disk that fully encloses any of the previously identified CSF disks. Then, if any of the remaining local features is salient, the most salient one is selected as an additional CSF. This procedure is repeated until no new features are found, and always terminates in practice after a small number of steps (usually 1 or 2). It requires a measure of CSF overlap, used to determine when one CSF encloses another, and a measure of CSF saliency. Both measures are described next.

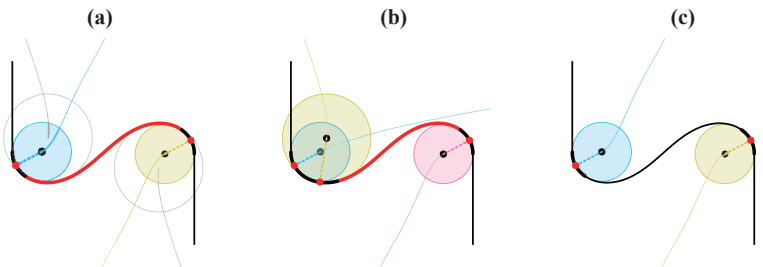


Figure 7.6: Overlapping disks along a spiral segment. **(a)** the segment in red between the contact regions of the two CSFs is a spiral. However its local medial axis has two branches producing two terminal disks, shown in gray. **(b)** Without filtering, the left disk produces an additional CSF, since it is slightly more salient than the other disk. **(c)** However, the disk fully encloses the previously identified one so it is discarded. This results in the spiral segment not producing any new CSF.

7.3.2 CSF Overlap

A spiral is a continuous curve with monotonically-varying curvature. Such a curve does not have any curvature extrema between its ends (Leyton, 1987) and thus should not produce an additional CSF. This can be further characterized by the *Tait-Kneser* theorem (Ghys et al., 2013), which states that all osculating circles of a spiral segment with strictly positive or negative curvature, are *disjoint* and *nested*. However, because CSF analysis operates on a sampled curve, looking for additional CSFs for an outline segment that closely resembles a spiral is likely to produce additional terminal branches in its local *SA* (Figure 7.6a). These branches can result in the detection of a CSF that does not correspond to an actual extremum (Figure 7.6b).

We discard such disks when evaluating additional CSFs by computing the *degree of overlap* $\delta_C \in [0,1]$ between any two discs as the area of the intersection between the disks divided by the area of the smaller disk. We discard any new terminal disk if there is a pre-existing CSF with a smaller disk radius and for which the degree of overlap for the disks is greater than a user-defined threshold, which we empirically set to 0.98 (Figure 7.6c).

7.3.3 CSF saliency

To measure the saliency of a CSF, we first extend its support segments to the extrema of adjacent CSFs, if present. These extensions are not used when finding CSFs because doing so could lead to less useful local axes (Figure 7.5c), but they capture a perceptually important region surrounding the CSF. We define the length, h, of the longest angle bisector coming from the extremum for any triangle connecting the extremum and two points, one on each extended support segment (Figure 7.7a). This usually occurs at the segment endpoints, but curved segments can sometimes lead to a maximal length before the ends (Figure 7.7c). The

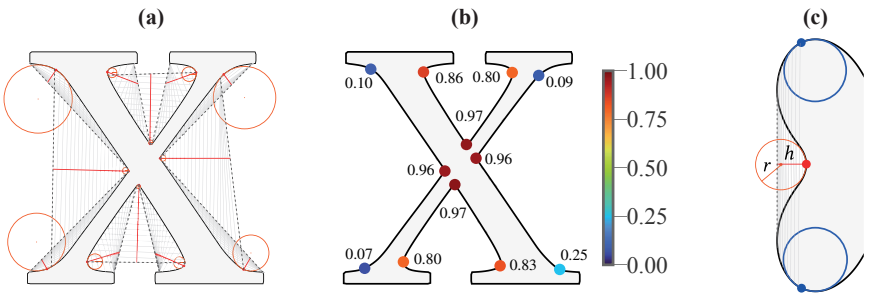


Figure 7.7: Concave CSF saliency computation for the outline of a glyph: **(a)** Concave CSFs and their triangles. **(b)** Saliencies for all concavities. Note that while saliency is somewhat correlated with the disk radius, two CSFs with similar radii, like the two leftmost ones, can have different saliencies because of the surrounding trace segments. **(c)** Detail of a saliency computation when the longest bisector h does not occur at the ends of the extended support segments; r is the radius of the disk for the concavity.

saliency of a CSF, c, is then given by:

$$w(c) = e^{-r/h} \quad , \tag{7.1}$$

with r being the radius of the associated terminal disk.

Saliency is meant to evaluate the curvature of a CSF, proportionally to the portion of 2D space "captured" by the CSF, as approximated by the selection of h and modulated by an exponential decay. In our experiments this measure is more robust than related outline-based saliency measures such as turning angle or "stick-out" (De Winter and Wagemans, 2008b). We consider a feature salient if w is above a small threshold, which we set to 1×10^{-3}, informed by the results in Chapter 8 and Chapter 10.

7.3.4 Computing the CASA

We construct the interior and exterior Curvature Augmented Symmetry Axes (CASA), denoted SA^I_+ and SA^E_+ , by combining SA^I and SA^E with parts of the local axes associated with CSFs. SA^I and SA^E are first augmented with segments that connect each terminal vertex to the extremum of the corresponding initial CSF. Then, we consider each *additional* CSF and add a new axis segment linking the corresponding curvature extremum to the first encountered intersection with the associated SA, thus creating a new fork (Figure 7.8). This process results, for closed contours, in local axis segments corresponding to convexities being added to SA^I_+ while those for concavities are added to SA^E_+(Figure 7.8).

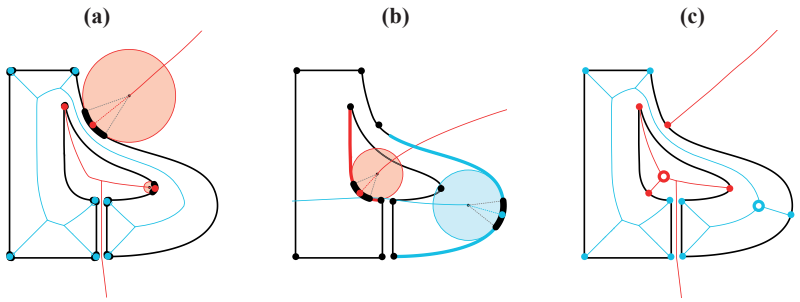


Figure 7.8: Retrieving CSFs and the CASA for a glyph outline. **(a)** SA^I (blue) and SA^E (red), and the initial CSFs found at branch terminals. Note that a concave ($m-$) and a convex ($M+$) CSF are missed because they do not occur at terminal branches of the traditional *SA*. **(b)** Local *SA*s are computed over the support segments highlighted in red (concave) and blue (convex), giving two additional CSFs. **(c)** The final interior (blue) and exterior (red) CASA, SA^I_+ and SA^E_+ resulting from identifying all salient CSFs. The two additional forks are emphasized with rings.

7.4 Absolute Curvature Minima CSFs with the ESAT

Most existing pattern recognition applications (Leymarie and Levine, 1988; De Stefano et al., 2005) as well as perceptual studies (Attneave, 1954; Feldman and Singh, 2005) are concerned with the identification and importance of *absolute maxima* of curvature. However, it has been shown that absolute minima of curvature are also often chosen as salient points along object boundaries (De Winter and Wagemans, 2008b). As mentioned in Section 3.8.2.3, the analysis of Leyton (1987) takes also these points into consideration with a variant of the SAT that he denotes as Exscribed Symmetry Axis Transform (ESAT), consisting of the loci of all minimally circumscribed disks to a contour segment (Figure 7.9.a). The ESAT disk with the maximal radius, is guaranteed to terminate at an absolute minimum of curvature if no other extrema are present along the same curve segment. In Blum's terminology the ESAT is a subset of the "unblocked" symmetry axis, or SS. Indeed we can see that with the computation of the ESAT of an ellipse, we recover its vertical *SA* (Figure 7.9).

7.4.1 Computing the ESAT: Farthest Voronoi Diagram

While Leyton's construction remains theoretical and limited to smooth contours, we have found a possible discrete implementation by considering higher order Voronoi diagrams, that is generalisations of the Voronoi diagram which consider the distance between multiple sites (Chazelle and Edelsbrunner, 1987). The order of the Voronoi diagram is given by the number of sites considered as closest, and thus the traditional (single) "nearest neighbour" Voronoi diagram is of order 1: each Voronoi region contains points nearest to a single input site. The n^{th} order Voronoi diagrams consists of regions in which points are nearest to n sites simultaneously. For an input made of n sites, the $n-1$th order Voronoi diagram is called the

Farthest (or Furthest) Voronoi Diagram (or FVD, (de Berg et al., 2008, Section 7.4)). A site will have an associated region in the FVD iff it is part of the convex hull of the input (Biedl et al., 2016). A contrario, sites within the convex hull have no regions in the FVD. The FVD *edges* define points which are equidistant (and farthest) from two generating sites and closer to all the others, Figure 7.9.c.

Circles centred along an edge and tangent to both sites contain all other sites. The same holds for FVD *vertices*, which are equidistant to three (or more) sites. The sites define a triangle, the circumcircle of which contains all the remaining sites and is centred at the FVD vertex. Similarly to the (1st order) VD case, this defines a triangulation that is dual to the FVD and which is known as the *Farthest Delaunay Triangulation* of the sites (de Berg et al., 2008) which is often used to compute the FVD (e.g. in the QHull package (Barber et al., 1996)).

This definition suggests a similarity between the FVD and the ESAT. For contour samples, the disks centered at the FVD edges and vertices touch the curve at 2 or more samples, and contain all the other samples of the curve in accord with the definition of the ESAT. This similarity can also be illustrated if we consider the formation of the FVD as the outcome of waves propagating from the input samples. Then, the edges of the FVD are the points at which two wavefronts collide, while interacting with all the remaining wavefronts. Assuming a constant velocity of propagation, two wavefronts meeting at a point implies equidistance between the two generating sites at the collision point, while the interaction with all other wavefronts implies that all the other sites are closer to the point than the two generating sites. This corresponds to Blum's model of "unblocked" symmetry axis (Blum, 1973) and is a subset of the SS (Giblin, 2000).

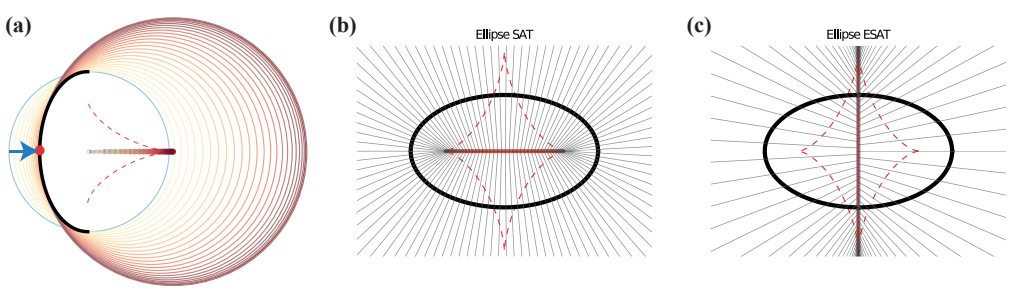


Figure 7.9: (a) ESAT of the portion of an ellipse and the corresponding circumscribed circles. The blue arrow indicates the corresponding PISA axis (or process arrow) which traces the midpoints of the arcs defined between the generating points (Leyton, 2012). The arrow defines a compression of the shape from a circular arc. **(b)** SAT and **(c)** ESAT of an ellipse. In dashed red is shown the evolute of the ellipse, and in gray the nearest (b) and farthest (c) Voronoi diagrams of point samples along the ellipse boundary.

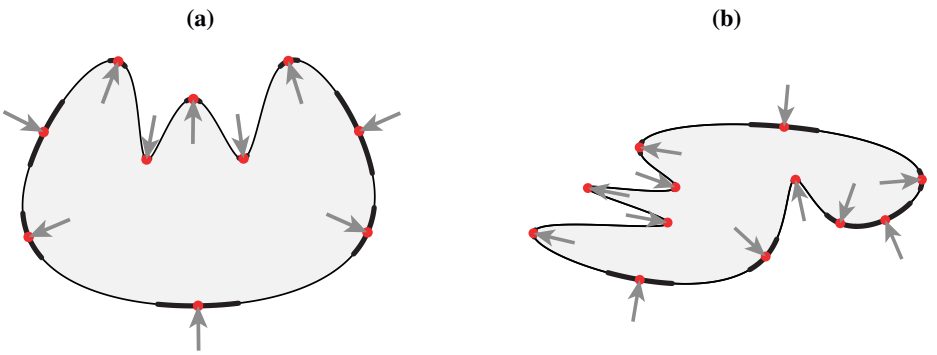


Figure 7.10: Contact regions (thick black segments) for absolute maxima and minima (red dots) of two smooth contours together with an approximation of Leyton's 1988 "process arrows" (gray) indicating the directions along which a supposed generative process has acted to produce the extrema. Note that , consistently with Leyton (1988), the process arrows for the absolute minima (all $M-$ in this case) are external to the shape interior, indicating a "squashing" process on the outline.

7.4.2 Identifying $m+$ and $M-$ CSFs

The observations above, lead us to conjecture that a subset of the FVD approximates the ESAT and that results similar to the nearest VD and SAT hold (Ogniewicz, 1992). However, computing the ESAT via the FVD is subject to sensitivity issues that are similar and apparently more severe than the SAT case. We leave research on a generalised solution to such sensitivity issues for future research. However, the identification of CSFs for absolute minima is feasible, since the computation is limited to support segments that can be smoothed without corrupting otherwise important trace features.

We note that a support segment can contain an absolute minimum if its adjacent CSFs have the same sign (or are curving towards the same side). As a first step we compute the mean squared error (MSE) for a straight-line fit to the support segment. If the MSE is less than a (user-specified) threshold, we can either ignore the minimum, or assume that it is located at the center of the (nearly flat) support segment. Otherwise, we compute the FVD for that segment. We reduce noise sensitivity issues by smoothing the segment with conventional smoothing spline (Dierckx, 1975), again with a user defined threshold. We take as discrete ESAT disk centers the mid-points of FVD edge that do not extend to infinity; then the disk radii are given by the corresponding distance to the pair of (farthest) generating sites. We then select the ESAT disk with maximum radius as the representative disk (Figure 7.9.a, largest red circle). The disk produces a CSF if it is salient according to the same procedure as described in Section 7.3.3, but using a lower saliency threshold of 1×10^{-6} for the examples given.

A linear approximation of the local *SA* for the CSF can be computed with the segment going from the disk center to the extremum. A similar approximation can be used to compute a set of "process arrows", similar to the ones defined by Leyton (1988) to indicate the directions along which a likely generative process has acted to produce an extremum (Figure 7.10). While in Leyton's theory these arrows correspond to the PISA, its computation in practice is highly sensitive to noise. Here we approximate the process arrows with vectors ending at the CSF extremum $\hat{\boldsymbol{z}}_i$For absolute maxima ($M+, m-$: protrusion and indentation) each vector is oriented opposite to the bisector of two tangents computed along the CSF support segments $\boldsymbol{z}_i^{lhs}(s)$ and $\boldsymbol{z}_i^{rhs}(s)$ and starting from the endpoints of the contact region $\overrightarrow{CC}_i$. For absolute minima ($M-, m+$: resistance and squashing) each vector has the same orientation as the the vector going from the CSF disk center to the $\hat{\boldsymbol{z}}_i$. This result remains mostly of theoretical interest, and will require future investigations to develop or test robust and sufficiently accurate implementations. Note that we will use a similar construction for negative minima of curvature ($m-$) in Chapter 10 to assist the segmentation of font outlines into strokes.

7.5 Transition Segments and Inflections

The CSF identification procedure determines a set of curvature extrema, where each CSF approximates a circular arc along the contour, via its contact region. In the following step we reconstruct the contour segments $\boldsymbol{z}_i(s)$ not covered by contact regions with *transition segments* consisting of either straight lines or Euler spirals.[5] This is performed with the simplifying assumption that the segments are either straight or characterised by a linearly varying curvature function.

The transition segments can identify *inflections* (Figure 7.11.a) and result in a piecewise-linear approximation of the curvature function (Figure 7.11.b). We will use this representation in the following chapter to efficiently recover $\Sigma\Lambda$ parameters from a trace. At the same time, this kind of approximation is similar to Euler spiral decompositions used for curve fairing applications (Baran et al., 2010; McCrae and Singh, 2009) and it is potentially useful for a similar task.

7.5.1 Fitting Euler Spirals

Recall from Chapter 4 that an Euler spiral, parameterized by arc length, is computed in terms of the Fresnel integrals $C(u), S(u)$ (equation (4.10)) and evaluated with:

$$\boldsymbol{q}(u) = (x(u), y(u)) = (C(u), S(u)) \quad , \tag{7.2}$$

where u can vary from minus to plus infinity, and where the origin, $(0,0)$, corresponds to $u = 0$, which is the *inflection* point for the spiral; note that u is then identically the (signed)

[5] This use of Euler spirals is also inspired by the work of Leyton who studied contour regions between curvature extrema of opposite sign by using "bi-spiral" segments (Leyton, 1987).

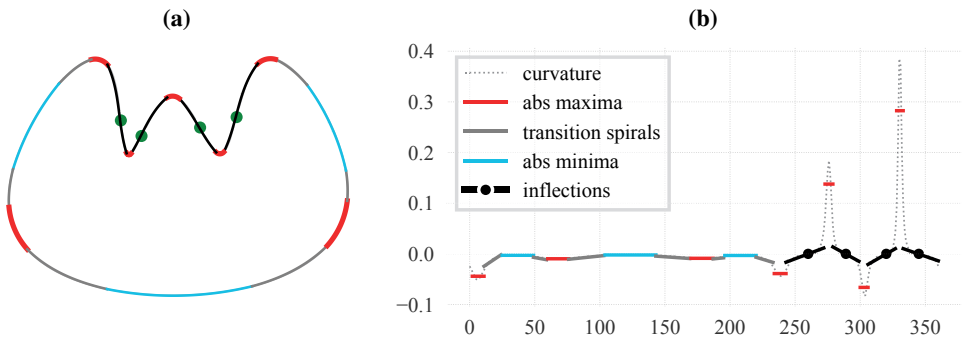


Figure 7.11: Reconstruction and curvature function approximation of a B-spline contour. **(a)** The contour reconstructed as a combination of circular arcs (CSF contact region) and Euler spiral segments. Black segments have an inflection (green circles). **(b)** The features overlaid as segments on the curvature of the input path. We can observe that the method gives a good approximation of the curvature function in correspondence with each extrema, as well as an estimate of the region along which curvature is approximately constant. The curvature estimation is less precise for points with high absolute curvature, due to the (relatively low) sampling frequency of the input and to a smoothing effect given by pruning of the skeleton with the chord residual. An adaptive sampling strategy would improve the fit.

arc length parameter for the spiral curve. In our application, an Euler spiral segment to be fitted to the data is defined between an initial ($u = u_1$) and a final ($u = u_2$) parameter values. If the values alternate in sign, then we have a segment with an inflection (at $u = 0$). Similarly to Chapter 4, such a segment can be sampled in an efficient manner using the method of Heald (1985), which results in $\tilde{N}$ samples along the segment going from $\boldsymbol{q}(u_1)$ to $\boldsymbol{q}(u_2)$.

In order to fit an Euler spiral segment to one of the segments from our input trace, we first compute approximate tangent directions along the trace, for a given support segment $\boldsymbol{z}_i(s)$, i.e. in correspondence with the initial and final points of the segment under examination. This allows to rapidly compute a first estimate of the spiral segment's initial and final parameter values using the same method by Levien (2009a) that we used in Chapter 4.

However, the tangent estimates are likely to be unreliable in the presence of noisy input data, and thus we proceed to refine this initial fit with a least squares optimisation based on the classic Gauss-Newton method. Our method consists then in three additional steps. First, we linearly transform the given support segment, $\boldsymbol{z}_i(s)$, such that its end points match those of the computed spiral segment in its canonical form. Second, we modify the canonical form

of the Euler spiral, by introducing a scaling factor a and a rotation by an angle ω with:

$$\boldsymbol{q}(u) = \begin{bmatrix} a\cos(\omega)C_1(u) - \alpha\sin(\omega)S_1(u) \\ a\sin(\omega)C_1(u) + \alpha\cos(\omega)S_1(u) \end{bmatrix}, \quad \text{where} \tag{7.3}$$

$$C_1(u) = C(u) - C(u_1) \quad \text{and} \quad S_1(u) = S(u) - S(u_1) \quad . \tag{7.4}$$

Note that the (initial) canonical form is for $a = 1$ and $\omega = 0$. Third, and finally, we proceed with the minimisation:

$$\min_{u_1,u_2,a,\omega} \frac{1}{2} \sum_{j=1}^{\tilde{N}} \| \boldsymbol{q}[j] - \boldsymbol{z}_i[j] \|^2 \quad , \tag{7.5}$$

where $\boldsymbol{q}[j]$ and $\boldsymbol{z}_i[j]$ both denote $\tilde{N}$ equally spaced samples with a sampling index $[j]$, in the former case for the spiral segment $\boldsymbol{q}(u)$ between u_1 and u_2, and in the later case along the input support segment $\boldsymbol{z}_i(s)$.

Figure 7.12: An Euler spiral, its inflection point (circle) and a Euler spiral segment (thick black).

7.5.1.1 Subdivision

The method above does not perform well when the support segment between two contact regions approximates spiral segments with a relatively high total turning angle (Figure 7.13a). This rarely happens when processing handwritten traces, but it can occur with other kinds of vector input.

Ideally, this could be solved by using a more flexible primitive to describe transitions, for example using the general aesthetic curve method of Miura (2006). Another less efficient method to overcome this issue is to subdivide a segment when the fitting error is greater than a given threshold. However, with the assumption that a support segment is always a spiral, we can efficiently determine if subdivision is necessary by computing the sum

$$|\phi| = \sum_s |\phi(s)|$$

of the *absolute* turning angles $|\phi(s)|$ along the support segment $\boldsymbol{z}_i(s)$ under consideration and where the absolute value is necessary because the segment may contain an inflection. We then recursively subdivide a segment in half if $|\phi|$ is greater than a user define threshold

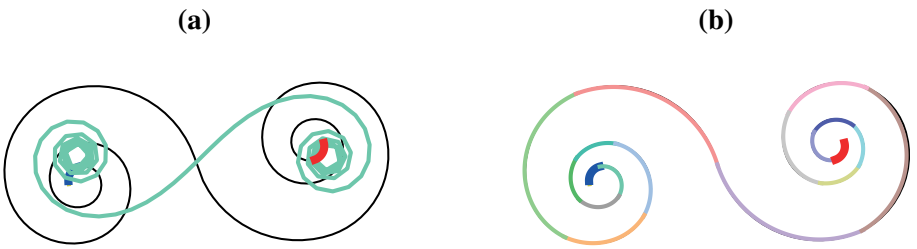


Figure 7.13: Subdivision of support segments for fitting Euler spirals. **(a)** The contour segment consists of two concatenated spiral segments with a high number of revolutions, and it produces two CSFs (red and blue) near the extremities. The support segment between the two CSFs is not an Euler spiral, which results in a poor fit. **(b)** Recursively subdividing the support segment results in a precise fit consisting of multiple transition segments.

(Figure 7.13b). We find that a threshold of $\frac{\pi 4}{5}$ works well for the use case of ΣΛ parameter reconstruction discussed in the next chapter, and we use the same threshold in the other examples given. We optionally smooth the support segment first using a convolution with a Gaussian, to avoid potential issues that can arise due to noise.

7.5.2 Inflections

The presence of inflections can be identified by checking if the two CSFs adjacent to a transition segments have absolute maxima of curvature of opposite sign. If this is the case we distinguish between three sub-cases. Similarly to the case of minima, we first check if the support segment $\boldsymbol{z}_i(s)$ is sufficiently straight, by testing if it can be approximated with a straight line with linear least square fit. If the fit MSE is less than threshold, we label the segment as straight and use its midpoint as representative of the inflection. Otherwise, if the support segment consists of a single transition segment, the location at which the spiral parameter $u = 0$ gives the inflection position. If a support segment is subdivided into multiple spirals, we check if successive spiral parameter pairs u_1 and u_2 of any given spiral have different signs, in which cases we can identify an inflection again with $u = 0$. If none of the spirals have alternating signs, we repeat the procedure for adjacent spirals using the parameter u_1 of the first and the parameter u_2 of the second. If the parameters have alternating signs, the inflection is located at the point where the two spirals meet.

7.6 Discussion

The CSF analysis procedure is written in the Python programming language, and relies on the QHull library (Barber et al., 1996) to efficiently compute 2D Voronoi diagrams used for the SAT recovery. For inputs consisting of closed contours (Figure 7.14a), the procedure takes

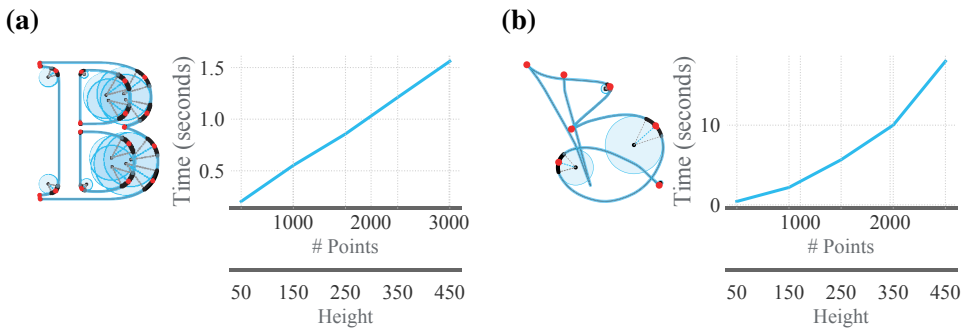


Figure 7.14: CSF computation performance for **(a)** closed contours and **(b)** open traces (with self-intersections) using a constant sampling distance of $\Delta s = 1$, increasing bounding box height and thus an increasing number of trace samples.

in average 0.5 seconds for the predefined bounding box height $h_{ext} = 150$ (Section 7.2.1). For inputs consisting of open traces (Figure 7.14b), the procedure is considerably slower and it takes in average 2 seconds for the same predefined bounding box height. This is due to the time complexity of the brute-force intersection method (Section 7.3.1), which could be improved with a sweepline-based approach (de Berg et al., 2008). At the same time, the intersection code is also written in Python, which is especially slow when performing loops (Behnel et al., 2011). As a result, a C/C++ implementation of that part of the code is likely to produce a significant performance gain.

We have tested our method on various inputs ranging from object silhouettes, font outlines to handwriting and graffiti traces. In the current implementation, the discrete Voronoi diagram is highly sensitive to circular or nearly-circular outlines, which can give results that vary depending on the scale, sampling frequency (Figure 7.15b) or quality (Figure 7.15c) of the input. In our experiments, this potential weakness does not have a serious adverse effect. Still, a robust method for circle, ellipse, and oval detection, such as the Hough transform (Manzanera et al., 2016), could identify these symmetric features, and be combined with our approach. Our method can also function with relatively noisy inputs, as long as an appropriate Voronoi SAT regularisation threshold is chosen (Figure 7.15.c). Currently, this threshold must be set by a user, but an automatic method is a useful avenue of future research.

We perform an approximate evaluation (Figure 7.16) of our method on stimuli taken from the dataset developed by De Winter and Wagemans (2004). The dataset contains the contours for 260 “everyday” object silhouettes (Snodgrass and Vanderwart, 1980), together with salient points along the object contours selected by approximately 40 participants per stimulus. Following De Winter and Wagemans (2004; 2008b), we compute a frequency for

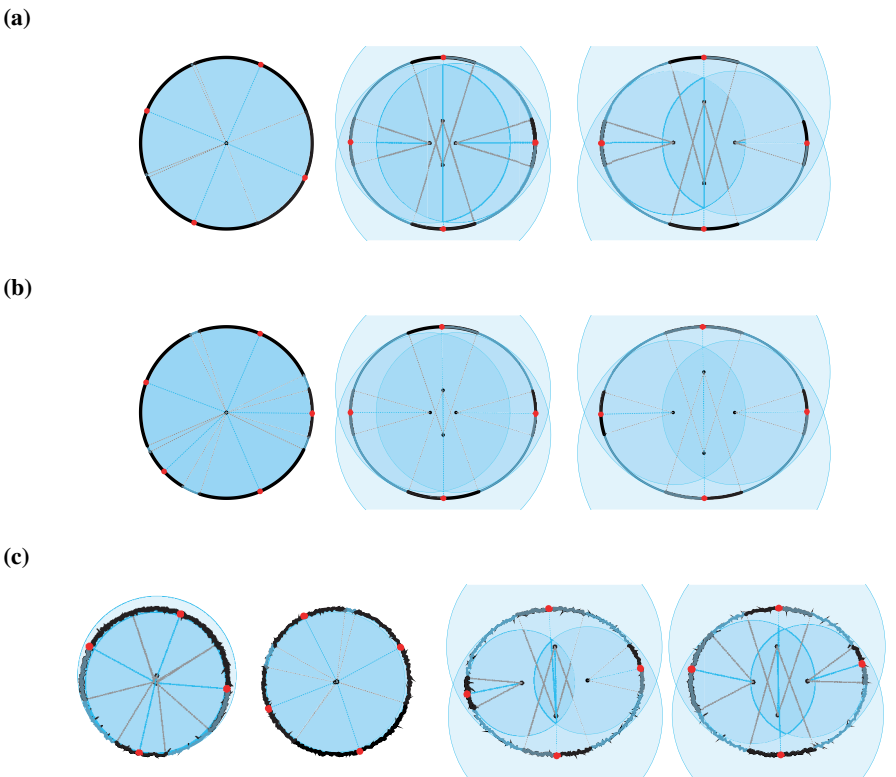


Figure 7.15: From left to right, CSFs for a circle, and two ellipses with increasing major axis size. **(a)** With high quality (h_{ext} = 150) contour samples, the location of the CSFs for the circle is not well defined, but the extrema for the ellipses are identified. **(b)** Slightly decreasing the bounding box height (h_{ext} = 100) modifies the CSFs for the circle (left), but the extrema of the ellipses remain stable. **(c)** With the addition of high-frequency noise, a relatively stable computation of CSFs for an ellipse is still possible, by increasing the Voronoi SAT regularisation threshold. However, the circle CSFs become unstable and their location and number depends on high frequency contour details.

each consecutive contour sample along an object outline. Each frequency is the number of times participants selected a contour sample as salient. We then compute a frequency-based saliency value for each contour sample by convolving the frequencies with a Gaussian. This will smooth out the noise potentially produced by participants selecting different but nearby contour samples. Finally we visually compare the frequency-based saliency values with the saliency (Section 7.3.3) computed for absolute minima and maxima CSFs along the same contour.

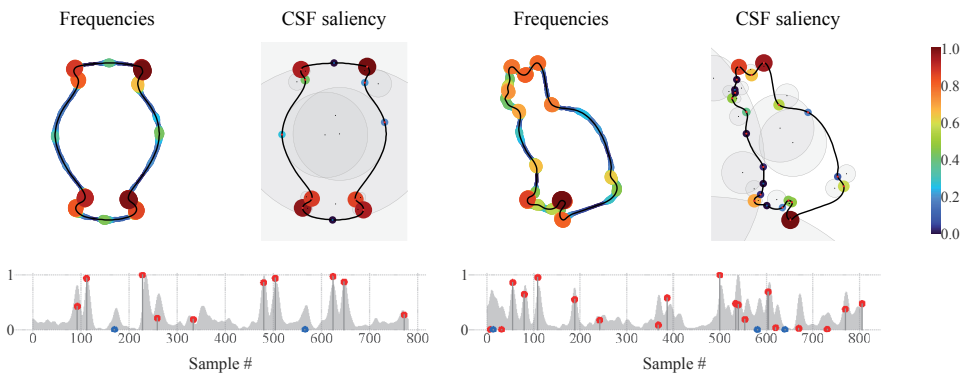


Figure 7.16: Qualitative comparison of salient points labelled by participants in the large-scale experiment of De Winter and Wagemans (2008b), and curvature extrema identified with CSFs. The top row shows the (smoothed) frequencies with which participants selected points along the outline (De Winter and Wagemans, 2008b) next to the saliency values computed according to the measure in Section 7.3.3. Both measures are normalised and colored according to the heat-map displayed on the right. Below each object, plot of the normalized frequency values for each points (grey), and the normalized saliency for the CSFs along the object outline displayed as red (absolute maxima) and blue (absolute minima).

Observing the plots in the second row of Figure 7.16 shows that the location of the CSF extrema along the outline (blue and red circles) generally corresponds with peaks in the smoothed frequency plot (gray). However, the normalized CSF saliency tends to be lower than the normalized frequency-based saliency. This is especially true for absolute minima of curvature ($m+, M-$, blue circles) and suggests room for improvement in the proposed CSF saliency measure, at least for the case of absolute minima. In future studies, we plan to perform a more rigorous analysis of the correlations between our saliency measure and the choices of participants, comparing it to other measures such as turning angle (Feldman and Singh, 2005; De Winter and Wagemans, 2008b) and in order to optimise our measure so it maximises consistency with experimental data.

7.7 Conclusion

In this chapter we have developed the notion of Curvilinear Shape Feature (CSF), a mixed boundary and region representation that is based on the computation of *local* symmetry axes. CSFs reconstruct contours or traces in terms of circular arcs and spiral segments and identify salient curvature extrema, together with a precise estimate of their associated center and radius of curvature. Each CSF is also paired with two support segments, that describe the region of influence of the feature and facilitate the estimation of local contour features such as tangents near concavities. For the case of closed contours, CSFs can be used to derive

the CASA: an augmented version of the SAT which has branches terminating at all curvature extrema and relating CSFs to the topology of an object.

We will use a number of these properties and representations to drive the methods presented in the following chapters. For example, in the next chapter we will use the extrema and the transition segments to derive a motor plan and circular arcs that reconstruct a trace in terms of $\Sigma\Lambda$ parameters (Figure 7.17). The support segments and the CASA will be useful in Chapter 10 (Figure 7.18) to relate CSFs with a measure of good continuation and to segment 2D font outlines into potentially overlapping and crossing strokes.

The implementation of CSFs discussed in this chapter opens up a number of avenues of future research. As previously mentioned in Section 7.5, CSFs allow a piecewise-linear approximation of the curvature function, which is potentially useful for curve-fairing applications (Baran et al., 2010; McCrae and Singh, 2009). CSFs also capture salient curvature extrema, so using saliency to determine the level of detail of a curve fairing procedure is an interesting research avenue. In the discussion of Section 7.6, we have seen that CSFs capture salient points that are consistent with the ones that are chosen by humans. However, our current choice of a saliency measure in Section 7.3.3 is principally driven by an evaluation of its performance during the implementation stage. A systematic comparison of different saliency measures with the data collected by De Winter and Wagemans (2008b) is of interest from a perceptual standpoint, but also useful to potentially improve the performance of our measure. Finally, we have seen in Section 7.4 how the FVD can be used to identify absolute minima of curvature and to produce a discrete symmetry axis representation that is similar to Leyton's ESAT (Leyton, 1987). While our proposed solution works in practice, an in-depth analysis of the relations between the ESAT and FVD would be not only useful to improve the robustness of our method, but also interesting from a theoretical perspective.

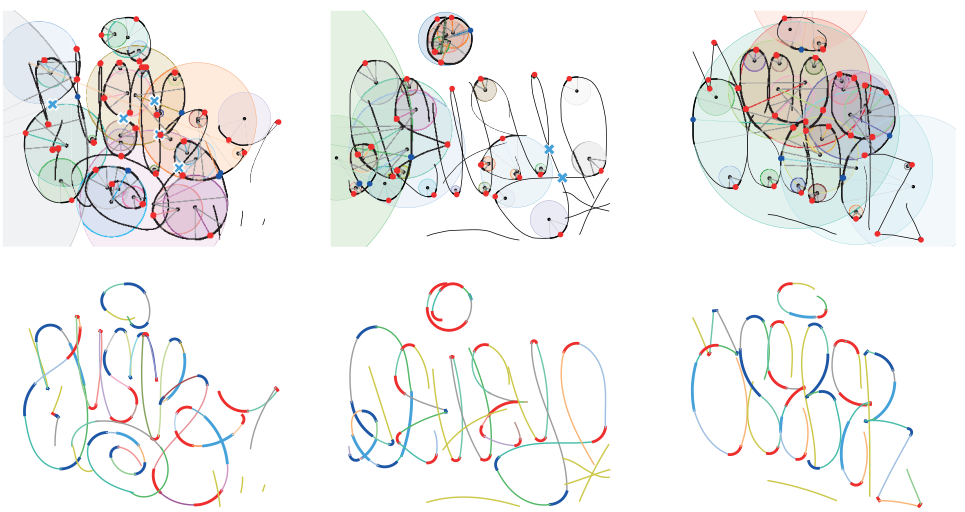


Figure 7.17: CSFs and transition segments of tag traces from the Graffiti Analysis database (Roth et al., 2009). *Top row:* CSFs and inflections. Absolute curvature maxima and minima are denoted with red and blue circles respectively. Inflections with blue crosses. *Second row:* Trace reconstruction with circular arcs (thick red and blue arcs) and Euler spiral transition segments (alternating colors).

Figure 7.18: (a) Interior and exterior SAT for a number of glyphs. **(b)** The corresponding CSFs.

Chapter 8

From Geometry to Kinematics with CSFs

This chapter is based on a collaboration between myself, Prof. Frederic Fol Leymarie, and Prof. Réjean Plamondon at Polytechnique Montréal (Canada), and is mostly based on a conference publication (Berio et al., 2018b) and a book chapter (Berio et al., 2020b).

In this chapter, we exploit the structure provided by CSFs to recover a *plausible* generative movement from the geometry of an input trace. By plausible, we mean a generative movement which can approximate closely that which is produced by the human hand, or even be mistaken as such. The method operates on inputs that can be represented as *ordered sequences of points*, such as digitised traces of handwriting, graffiti, drawings or the curves of vector art inputs. The procedure relies on a geometric analysis of a trace to automatically produce a *motor plan* and a set of *kinematic parameters*, the combination of which results in a *kinematic realisation* that reproduces the trace. This effectively extends the previously discussed concept of "style as kinematics" to trace inputs and enables meaningful variations and stylisations that would be difficult to achieve by relying on the trace geometry only. In particular, the kinematic parameters are expressed in terms of the $\Sigma\Lambda$ model, thus enabling the user-interaction, rendering, animation and parametric variations techniques already discussed in Chapter 4 to trace inputs. The $\Sigma\Lambda$ model is particularly well suited for this task, since it describes both a plausible movement and the resulting trace with a sequence of well defined primitives: ballistic sub-movements each characterised by a set of parameters that can be systematically modified to generate meaningful variations and stylisations of the trace.

A number of methods already exist that recover $\Sigma\Lambda$ parameters from digitised traces of handwriting (O'Reilly and Plamondon, 2008; Plamondon et al., 2014; Fischer et al., 2014; Ferrer et al., 2018). However these methods rely on an analysis of the input velocity and are aimed at biometric or pattern recognition purposes. For the scope of this thesis, our aim is

rather perceptually and artistically driven: to *infer a physiologically plausible motion* from an input trace, where the kinematics of the input may be *unavailable*, such as when using vector graphics inputs, or may be degraded or *unreliable* due to the poor quality of a digitisation device, such as when using low cost tablets or trackpads. As a result, we purposely ignore any pre-existing kinematics encoded by the input, in order to seamlessly handle any vector input in which only the sequential ordering of points may be available. We also choose this approach with the future aim of combining our method with one that recovers temporal information from bitmap images such as the one presented by Plamondon and Privitera (1999).

In the following sections, we first describe how the previously identified CSFs and transition segments can be transformed into a series of circular arcs (Section 8.1) that are used to drive the reconstruction procedure (Section 8.2). The arcs determine an initial estimate of a motor plan and a corresponding set of $\Sigma\Lambda$ parameters (Section 8.2.1), which are then refined to accurately reconstructs the input trace (Section 8.2.2). Section 8.3 demonstrates how this reconstruction can be used to create variations and stylisations of a trace with the same methods previously described in Chapter 4 and with extensions to these methods that further exploit the reconstruction procedure. These variations can be used to generate and render graffiti in a Procedural Content Generation (PCG) setting. Finally, in Section 8.4 we use the reconstruction procedure to compare the $\Sigma\Lambda$ model to MIC and to the minimum jerk (MJ) model , in particular to its path-constrained formulation (Todorov and Jordan, 1998), which is also capable of inferring physiologically plausible kinematics given an input trace.

8.1 Segmentation method

The proposed trajectory reconstruction method exploits the prior feature analysis of the input $\boldsymbol{z}(s)$ (Chapter 7), and thus takes as its input a set of CSFs and a set of Euler spiral transition segments. The CSFs consisting of a set of N_{CSF} terminal disks, CC_i, circular arc contact regions, $\overrightarrow{CC}_i$, curvature extremum loci, $\hat{\boldsymbol{z}}_i$, and $N_{CSF}+1$ support segments $\boldsymbol{z}_i(s)$. Adjacent CSFs share one support segment ($\boldsymbol{z}_i^{rhs}(s) = \boldsymbol{z}_{i+1}^{rhs}(s)$) and the first and the last CSFs have one unique support segment each ($\boldsymbol{z}_1^{lhs}(s)$ and $\boldsymbol{z}_{N_{CSF}}^{rhs}(s)$). Each support segment is approximated by one or more Euler spiral transition segments as described in Section 7.5.1.1.

8.1.1 Circular arc decomposition

On the basis of this information, the goal is then to segment the entire trace, $\boldsymbol{z}(s)$, with a series of best fitting geometric primitives. For generality, in this chapter we shall focus on circular arc primitives, since these are the basis for the conventional formulation of the $\Sigma\Lambda$ model. However, the method can easily be extended to Euler spirals for the case of the $\omega\mathscr{E}\Sigma\Lambda$ model (Section 4.2.2). The trajectory segmentation in terms of a series of circular arcs is executed with a systematic method, that starts from the Euler spiral transition segments and

approximates each such segment with one or two best fitting circular arc(s), as a function of the presence of an inflection along the spiral segment.

For the reconstruction task, we consider an inflection along a spiral valid only if the corresponding support segment has not been labelled as straight (Section 7.5.2) and the ratio of the spiral parameters $\min(u_1, u_2)/(u_1 + u_2)$ is greater than a user defined threshold, which empirically set to 0.2 in the accompanying examples. If the ratio is less than the threshold, we discard the inflection as a near degenerate case, the inflection being very close to one spiral's end point.

Depending on the presence of a valid inflection, each Euler spiral segment results in one or two circular arcs. The internal angle of the circular arcs is easily estimated by integrating the curvature of the spiral and distinguishing between 3 cases.

(a) For the case of two arcs, the internal angles are given by $u_1|u_1|$ and $u_2|u_2|$ (Figure 8.1a).

(b) In the case of a degenerate inflection, we use the same method to fit a single arc and choose only the parameter with the greatest absolute value ($|u_1|$ or $|u_2|$) and consequently higher curvature (Figure 8.1b).

(c) When no inflection is present (Figure 8.1c) the internal angle is given by:

$$|(u_2|u_1| - u_1|u_1|)|\operatorname{sgn}(u_1).$$

In summary, we have that the trace is now represented by a sequence of N_{CSF} contact circular arcs, $\overrightarrow{CC}_i$, with intermediate transition Euler spiral segments each with or without

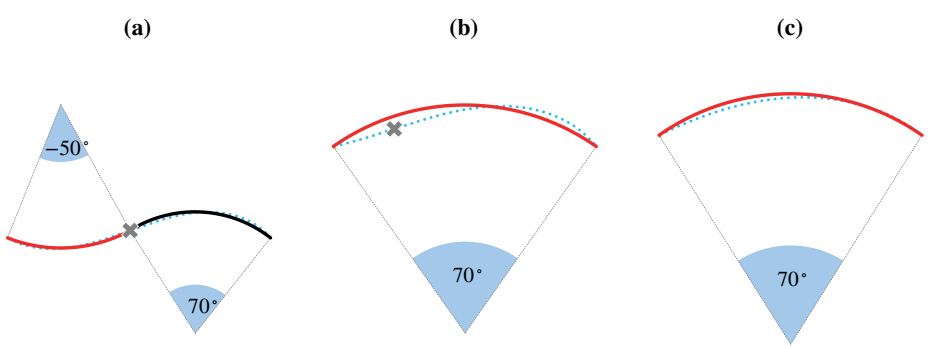


Figure 8.1: Decomposing Euler spirals (stippled cyan) into arcs. **(a)** two arcs delimiting an inflection (grey cross). **(b)** One segment with a degenerate inflection resulting in a single arc. **(c)** One without an inflection and resulting in a single arc.

an inflection. Each spiral segment is then mapped to either a pair of arcs (with a separating inflection) or a single arc.

Figure 8.2 shows the results of (i) identifying (here, five) curvature maxima, $\hat{\boldsymbol{z}}_i$, followed by (ii) fitting (five) Euler spirals, and (iii) finding (two) inflections and corresponding circular arcs. This approximate reconstruction of the original trajectory in the form of circular arc segments, is related to the method originally proposed by Li et al. (1998), but with the following main three differences: (i) We have found experimentally that our method to identify curvature maxima is more robust — in particular as it does not rely on an explicit *a priori* estimation of the curvature signal. (ii) We use Euler spirals to fit intermediate data which gives a simpler and more robust method to identify inflections. (iii) We explicitly obtain contact circle arcs, $\overrightarrow{CC}_i$, which results in a more accurate reconstruction of the original trace (Figure 8.2.c and d).

This representation is now ready to be exploited in the next section to iteratively reconstruct $\Sigma\Lambda$ parameters from the input trace, $\boldsymbol{z}(s)$.

8.2 Iterative Reconstruction of $\Sigma\Lambda$ parameters

Given the previous trace segmentation derived from identified CSFs and Euler spiral transition segments, we now have the necessary information to describe how we reconstruct the input trajectory with an approximate associated kinematics given only information about its (static) sampled geometry. Hence, we will be able to seamlessly process on-line handwriting data as well as vector art in which only the sequential ordering and coordinates of trace samples is required. The method is a development and improvement over our prior efforts (Berio and Leymarie, 2015; Berio et al., 2017a). We re-emphasise that, although a number of methods exist for the accurate reconstruction of $\Sigma\Lambda$ parameters from digitised traces (O'Reilly and Plamondon, 2008; Plamondon et al., 2014; Fischer et al., 2014), these require as input the explicit kinematics of the original trajectory.

8.2.1 Initialisation: Features, Sub-movements, Initial Targets

The initial virtual targets (i.e. the motor plan) consists of two types of *feature points*, or *features* for short: from CSF analysis (i) recovered curvature extrema, $\hat{\boldsymbol{z}}_i$, and from Euler spiral analysis and circular arc decomposition (ii) inflections or points where multiple circular arcs meet. We can either directly use these loci or find their nearest neighbors, $\boldsymbol{z}(\hat{s}_i)$, on the original input trace, $\boldsymbol{z}(s)$, which leads to slightly more accurate reconstructions. We follow the later approach in results reported hereafter.

An initial set of M *submovements* is defined from only those circular arcs derived from the Euler spiral segment fitting; i.e. we do not generate submovements for each contact circular arc, $\overrightarrow{CC}_i$, associated to each $\hat{\boldsymbol{z}}_i$. Furthermore, given the $\Sigma\Lambda$ modelisation, each $\overrightarrow{CC}_i$ is likely to coincide with a curved trace segment that is produced by the time superposition

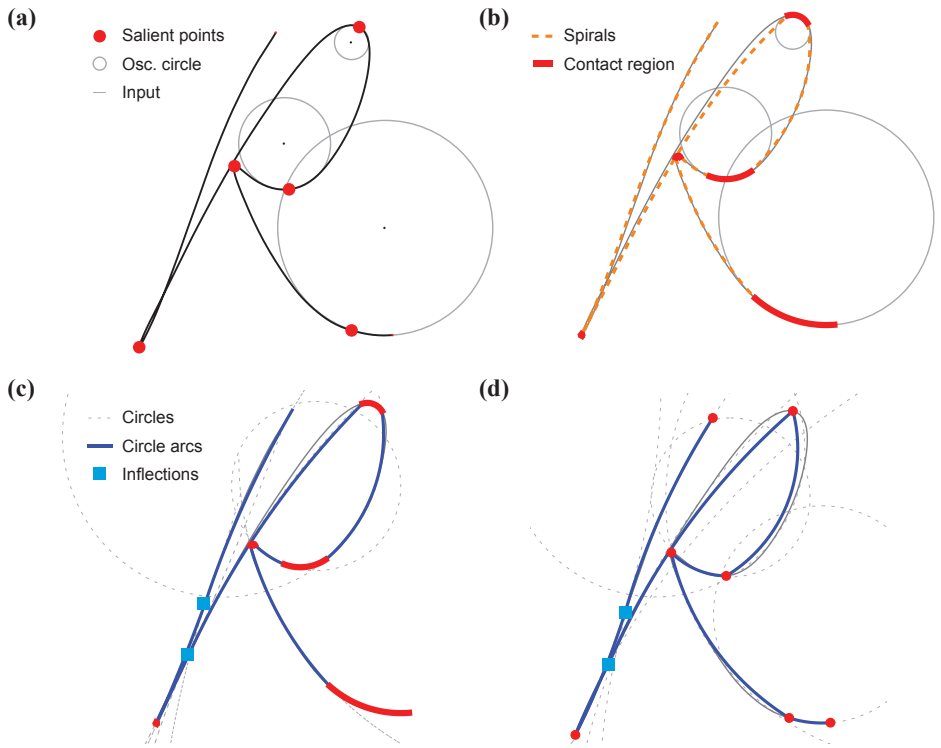


Figure 8.2: **(a)** Feature extraction based on CSFs, followed by **(b)** Euler spiral fitting, and **(c)** circular arc decomposition of a sample from the UJI handwritten character dataset (Llorens Piñana et al., 2008). The arcs in red indicate the intersections (with the original trace) of the circles of curvature corresponding with salient points. **(d)** Demonstrative example of least-squares fitting of circular arcs to the segments defined between consecutive salient points. Not considering the contact region results in less precise reconstruction of the input (compare **(d)** with **(c)**).

of two ballistic submovements, independently of the sub-movement curvilinear geometry. As a result, not considering these segments when estimating the ΣΛ circular arcs avoids the potential over-estimation of the corresponding curvature parameters δ_i (Figure 8.2.d).

The circular arcs we use — all derived from Euler spiral segments — give us a set of M internal angles $\hat{\theta}_i$, centers $\boldsymbol{c}(\hat{\theta}_i)$ and radii $r(\hat{\theta}_i)$. These arcs are delimited by $M+1$ feature points $\boldsymbol{z}(\hat{s}_i)$ with $\{\hat{s}_0, \hat{s}_M\}$ indicating the initial and final trace points. When modeling a closed trace or contour, we randomly pick one feature point as both the start and end positions. Each feature point corresponding to a CSF is also associated with an osculating circle with radius $r(\hat{s}_i)$, curvature $\kappa(\hat{s}_i)$ and center $\boldsymbol{c}(\hat{s}_i)$, as indicated by the corresponding terminal disk CC_i. The corresponding contact region $\widehat{CC}_i$ covers a portion of $\boldsymbol{z}(s)$ that is not covered by any of the previously identified circular arcs and at which the curvature $\kappa(\hat{s}_i)$ is approximately

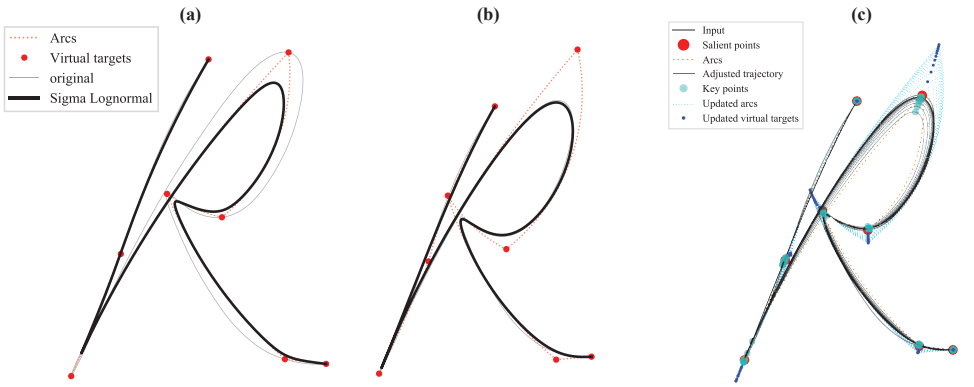


Figure 8.3: ΣΛ parameter reconstruction using features from CSFs and Euler spiral derived arcs. **(a)** First guess (in black) of the stroke parameters and motor plan from features. **(b)** Reconstruction of the input after iterative refinement steps. **(c)** Iterative refinement steps. The initial motor plan has targets corresponding with the features along the input (large red circles). At every iteration, the targets are shifted (small blue circles) in order to reduce the distance between keys along the generated trajectory (cyan circles) and the features of the original trace.

constant (Figure 8.2). Note that, in the vicinity of curvature extrema for which the trajectory is smoother, we obtain a larger radius of curvature (as expected), as well as a larger contact region.

An initial estimate of the trajectory is given by a motor plan with vertices $\boldsymbol{p}_i = \boldsymbol{z}(\hat{s}_i)$ and corresponding curvature parameters $\delta_i = \theta(\hat{s}_i)$. The time overlap parameters are initially set either (i) to $\Delta t_i = 0.5$ if a feature point $\hat{s}_{i-1}$ corresponds to an absolute maximum of curvature in $\boldsymbol{z}(s)$, or (ii) to a user defined minimum Δt_- otherwise — i.e. for inflection points, absolute curvature minima, or where any two subdivided transition segments meet. Similarly to the interactive use-case (Chapter 4) and for the sake of simplicity, the remaining parameters σ_i and μ_i are fixed to a user configurable value, with the assumption that they describe typical properties of the neuromuscular system of a writer. The initial trajectory estimate is likely to differ from the original, $\boldsymbol{z}(s)$, and to be much smoother due to the initial lognormal stroke overlaps (Figure 8.3.a).

8.2.2 Iterative scheme: Keys, Max speeds, Moving Targets

To improve the reconstruction, we adopt an iterative refinement scheme (Figure 8.3.c) in which we adjust the curvature and time overlap parameters together with the target positions in order to minimise the difference between the reconstructed trajectory $\boldsymbol{x}(t)$ and the input trace $\boldsymbol{z}(s)$. At each iteration, we rely on the estimation of a series of $M-1$ *key points* $\{\tau_i\}_{i=1}^{N_S}$, which approximate the initial feature point loci and are computed identically to Section 4.4.2.1. Recall that such key points indicate the time occurrence at which the influence of

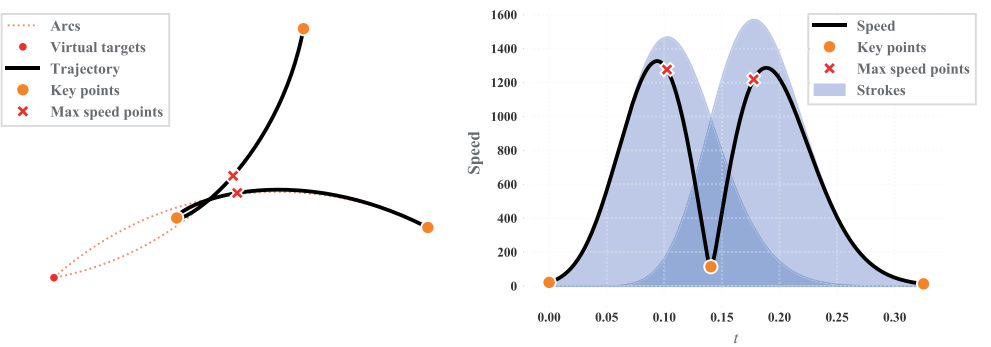


Figure 8.4: Key points (orange circles) and max speed points (red crosses) overlaid on (left) the trace and (right) speed profile of a trajectory made of two primitives. Note that the time occurrence of the "max speed points" coincides with the peaks of the lognormals (blue), but does not necessarily coincide exactly with the maximum trajectory speed. However, this approximation is simple and has proven to be sufficient for our reconstruction use case.

one lognormal exceeds the previous, which approximately corresponds to curvature extrema along $\boldsymbol{x}(t)$ (Figure 8.4).

In addition to key points, we also compute M *maximum speed points*, or *max speeds* for short, $\{\gamma_i\}_{i=1}^{N_S}$, which indicate the approximate time occurrence of the maximum speed for each stroke (Figure 8.4); this is explicitly obtained by the mode of the corresponding lognormal: $t_{0i} + \exp(\mu_i - \sigma_i^2)$.

8.2.3 Underlying observations

The iterative refinement scheme is designed based on three observations:

Observation 1. The time parameter Δt_i is proportional to the curvature $\kappa(\tau_i)$ at the time of the corresponding key point. Thus, a higher value of Δt_i will decrease the amount of overlap of successive lognormals. This will result in a lower speed and higher curvature $\kappa(\tau_i)$ at the time occurrence of the key. Since we have a good approximation of the curvature $\kappa(\hat{s}_i)$ in the original trajectory, the relation between the two can be exploited in order to adjust Δt_i proportionally at each iteration. We observe that changes in Δt_i are not linearly related to changes in the curvature $\kappa(\tau_i)$ at the corresponding key. In order to compensate for this, we assume a 1/3 power relation (Viviani and Schneider, 1991) which has often been observed in human movement and particularly holds for elliptical portions of the trajectory (Plamondon and Guerfali, 1998a), which is often the case near keys. The reasoning is that given the relations:

$$\Delta t \propto \kappa \quad \text{and} \quad \Delta t \propto 1/v \quad ,$$

where v denotes speed, we have the proportions relating desired and generated curvature and velocity:

$$\rho_\kappa = \hat{\kappa}/\kappa \quad \text{and} \quad \rho_v = \hat{v}/v \quad .$$

As a result, given the *power law* (Viviani and Schneider, 1991) $v = \kappa^{-1/3}$ and because velocity and Δt are inversely proportional, we finally get the relation:

$$\rho_{\kappa v} = v/\hat{v} = (\kappa/\hat{\kappa})^{-1/3} = (\hat{\kappa}/\kappa)^{1/3} \quad .$$

Observation 2. Moving targets play a role similar to control points in spline analysis. Shifting a target $\boldsymbol{p}_i$ in a given direction will cause the generated key $\boldsymbol{x}(\tau_i)$ to move in a similar direction. As a result, shifting the target $\boldsymbol{p}_i$ along the vector $\boldsymbol{z}(\hat{s}_i) - \boldsymbol{x}(\tau_i)$ will decrease the distance between successive generated keys and original features (Figure 8.3.(c)).

Observation 3. The distance D_i between successive targets $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i-1}$ will influence the curvature of the resulting stroke. Augmenting this distance will increase the radius of curvature of the circular arc defined by the parameter δ_i and will result in a decrease of curvature for the stroke. While the trajectory tends to depart from the circular arc near the keys at $t = \tau_i$ due to the smoothing effect of the lognormal time overlap, it tends to pass closer to the circular arc at $t = \gamma_i$ where the amplitude of the lognormal is maximal. As a result, we use this locus to evaluate the deviation from the desired arc $\hat{\theta}_i$ and correct the parameter δ_i accordingly.

Out of these three observations, we define each iteration of our method to consist of the following ordered steps:

$$\Delta t_i \leftarrow \Delta t_i + \lambda_\Delta(\varsigma(\hat{\Delta t_i}, \Delta t_{min}, \Delta t_{max}) - \Delta t_i) \quad , \tag{8.1}$$

$$\delta_i \leftarrow \delta_i + \lambda_\delta(\hat{\delta_i} - \delta_i) \quad \text{and} \tag{8.2}$$

$$\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i + \lambda_p(\boldsymbol{z}(\hat{s}_i) - \boldsymbol{x}(\tau_i)) \quad . \tag{8.3}$$

In the above formulation, we experimentally set the value of three damping parameters: $\lambda_\Delta = 0.1$, $\lambda_\delta = 0.1$ and $\lambda_p = 0.5$; these permit to avoid excessive adjustments at each iteration. The target time offset parameter for each iteration is computed by assuming a 1/3 power relation to curvature and is given by:

$$\hat{\Delta t_i} = \varsigma\left((\kappa(\hat{s}_i)/\kappa(\tau_i))^{1/3}, \Delta t_-, \Delta t_+\right) \quad , \tag{8.4}$$

which is restricted to a user specified range $[\Delta t_-, \Delta t_+]$ by using a logistic function:

$$\varsigma(x, a, b) = a + \frac{b-a}{1 + \exp\left(-20\left(x - \frac{a+b}{2}\right)\right)} \quad ,$$

with the multiplicative factor 20 empirically set to produce a steep logistic curve.

We observe that this restricted range improves convergence of our method and permits to apply smoothing effects to the trajectory during the reconstruction step (examples are given in Section 8.3).

The desired internal angle of an arc is given by:

$$\hat{\delta}_i = 4\tan^{-1}\left[\frac{h}{a}\tan\left(\frac{\delta_i}{4}\right)\right] \quad \text{with} \tag{8.5}$$

$$a = \| \boldsymbol{p}_i - \boldsymbol{p}_{i-1} \| \quad \text{and} \tag{8.6}$$

$$h = \left(r(\hat{\theta}_i) - \| \boldsymbol{x}(\gamma_i) - \boldsymbol{c}(\hat{\theta}_i) \|\right)\mathrm{sgn}(\hat{\theta}_i) \quad , \tag{8.7}$$

where the term h determines the amount to shift the curvature parameter δ_i by comparing the radius of the circular arc $\hat{\theta}_i$, initially fitted to the input, to the distance between its center and the lognormal max speed point $\boldsymbol{x}(\gamma_i)$.

8.2.4 Stopping Criteria, SNR

A few different stopping criteria for the iterative scheme are possible, depending on the user needs. The simplest — and most practical for experimenting with the approach — is to let the user define an overall maximum iteration. Other more sophisticated criteria we have experimented with include: (i) let keys reach each associated CC_i or $\widehat{CC}_i$; (ii) minimise the overall distance between the generated, $\boldsymbol{x}(t)$, and the input, $\boldsymbol{z}(s)$, traces, by either selecting a threshold value or letting the algorithm reach a local minimum; (iii) optimise the quality of the reconstruction by maximising an error criterion such as the SNR (defined next). We have found in practice the latter SNR-based criterion gives a good compromise between reconstruction quality and computational complexity.

Because we do not take into consideration the kinematics of the input, we evaluate the quality of the reconstruction using the Signal to Noise Ratio (SNR) computed between the reconstructed and input trajectory. While this could be done by uniformly sampling the two trajectories at a constant distance step, this will result in a propagation of errors along the reconstructed trajectory, which leads to unreliable SNR measurements. To overcome this problem, we exploit our initial estimation of features $\boldsymbol{z}(\hat{s}_i)$ in the input and the segmentation given by the keypoints τ_i of the reconstructed trajectory, $\boldsymbol{x}(t)$, and uniformly sample m segments for the original and generated trajectory, where the jth point for the ith segment are respectively denoted as $\boldsymbol{z}_{i,j}$ and $\boldsymbol{x}_{i,j}$ and the mean of an input segment is denoted by $\bar{\boldsymbol{z}}_i$. The trajectory SNR is then:

$$SNR = 10\log_{10}\frac{\sum_i\sum_j\left(\boldsymbol{z}_{i,j}-\bar{\boldsymbol{z}}_i\right)\cdot\left(\boldsymbol{z}_{i,j}-\bar{\boldsymbol{z}}_i\right)}{\sum_i\sum_j\left(\boldsymbol{z}_{i,j}-\boldsymbol{x}_{i,j}\right)\cdot\left(\boldsymbol{z}_{i,j}-\boldsymbol{x}_{i,j}\right)} \quad , \tag{8.8}$$

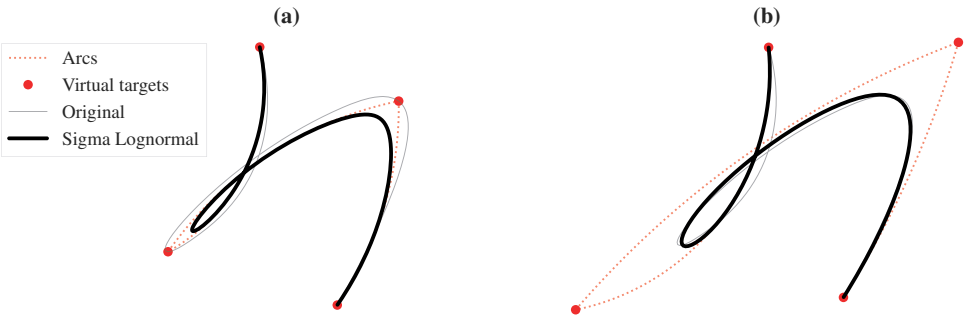


Figure 8.5: Reconstruction of vector input initially built with piecewise Bézier curves. Our method reconstructs the (originally only guaranteed to be C^0 continuous) input with smooth kinematics given by the $\Sigma\Lambda$ model. **(a)** First guess of the parameters from features. **(b)** Reconstruction of the input after iterative refinement steps.

which easily generalises to the case of multiple disconnected trajectory segments, such as when the writer lifts-up their pen or brush.

We tested the iterative refinement on different inputs ranging from vector traces with no a priori kinematic information (Figure 8.5), to online data — including the Graffiti Analysis database (Roth et al., 2009) (Figures 8.6, 8.7, 8.8 and 8.9) and the UJI handwritten character dataset (Llorens Piñana et al., 2008) (Figure 8.3) — and it consistently produces visually accurate reconstructions of the input. We observe that, while fluctuations may appear during iterations, the refinement scheme consistently and rapidly converges towards a reduction of the error between the input and the generated trajectories and an increase in SNR (equation (8.8)).

The iterative scheme can be applied in a *batch* manner, in which all the $\Sigma\Lambda$ parameters for all strokes are updated at each iteration, or similarly to the iDeLog framework (Ferrer et al., 2018) by traversing the trajectory in an *incremental* manner and adjusting pairs of strokes ordered in time. In our experiments both approaches present similar convergence properties and produce reconstructions with similar SNR.

8.3 Editing, Rendering and Stylistic Variations

The output of the reconstruction procedure, extends all the functionalities demonstrated in Chapter 4 to arbitrary traces. This enables a user to fine-tune the rendering results or to apply subsequent modifications to the trajectory by editing the target positions and the primitive parameters through a CAD-like interface. The resulting kinematics reproduce natural human-like movements that can be exploited to create primitive animations of the input as well as to generate smooth motion paths for virtual characters or even humanoid robots (Be-

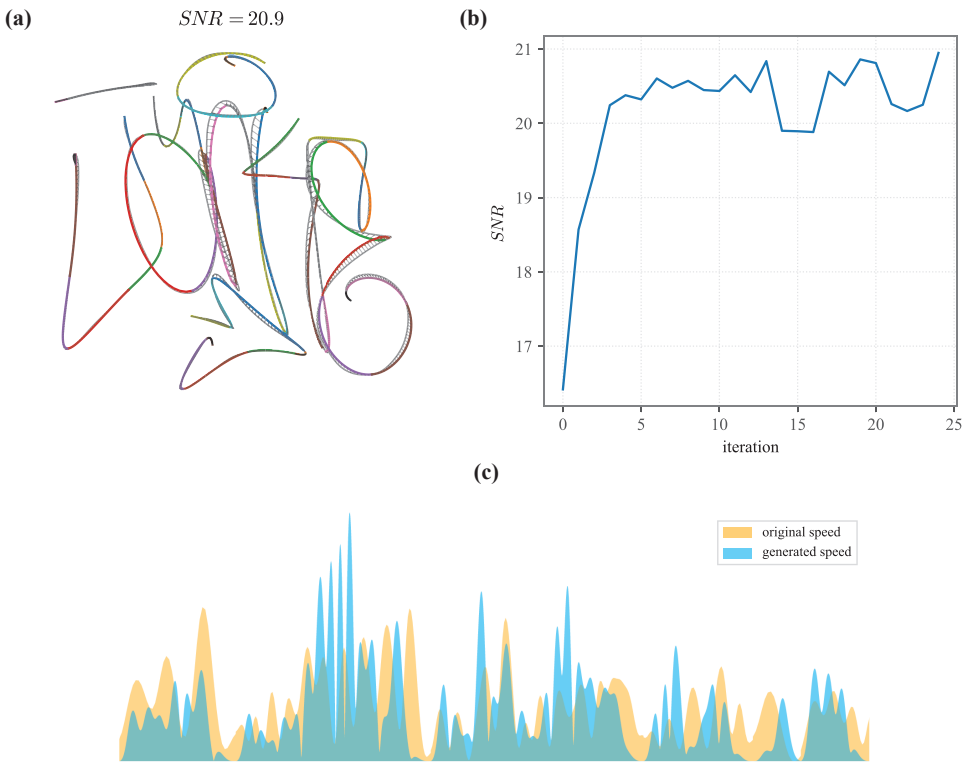


Figure 8.6: Reconstruction of a graffiti signature "JANKE" from the Graffiti Analysis database (Roth et al., 2009). **(a)** The reconstructed trajectory, subdivided into segments for comparison (color coded), overlaid on the original trace (light grey). The short grey segments mark the errors and correspondences between uniformly distanced samples for each trajectory segment. **(b)** Plot of SNR_t for each iteration of the iterative optimization scheme. **(c)** The speed profiles of the original (light grey) and reconstructed (dark grey) trajectories, scaled for comparison.

rio et al., 2016).

8.3.1 Smoothing and Fairing.

As seen in Chapter 4, smoothing (or its opposite, “sharpening”) effects can easily be achieved by globally scaling the Δt_i parameters. Combined with parameter reconstruction, this results in a procedure that bears similarities to computer graphic approaches for curve fairing or neatening (Thiel et al., 2011; McCrae and Singh, 2009) as well as curve stylisation approaches (Lang and Alexa, 2015; Lu et al., 2012). In particular, the Euler spiral decomposition step of the reconstruction method is similar to some previously proposed methods (Baran et al., 2010; McCrae and Singh, 2009), which exploit the decomposition of an input curve into Euler spiral segments to remove discontinuities and guarantee C^2 continuity in the output.

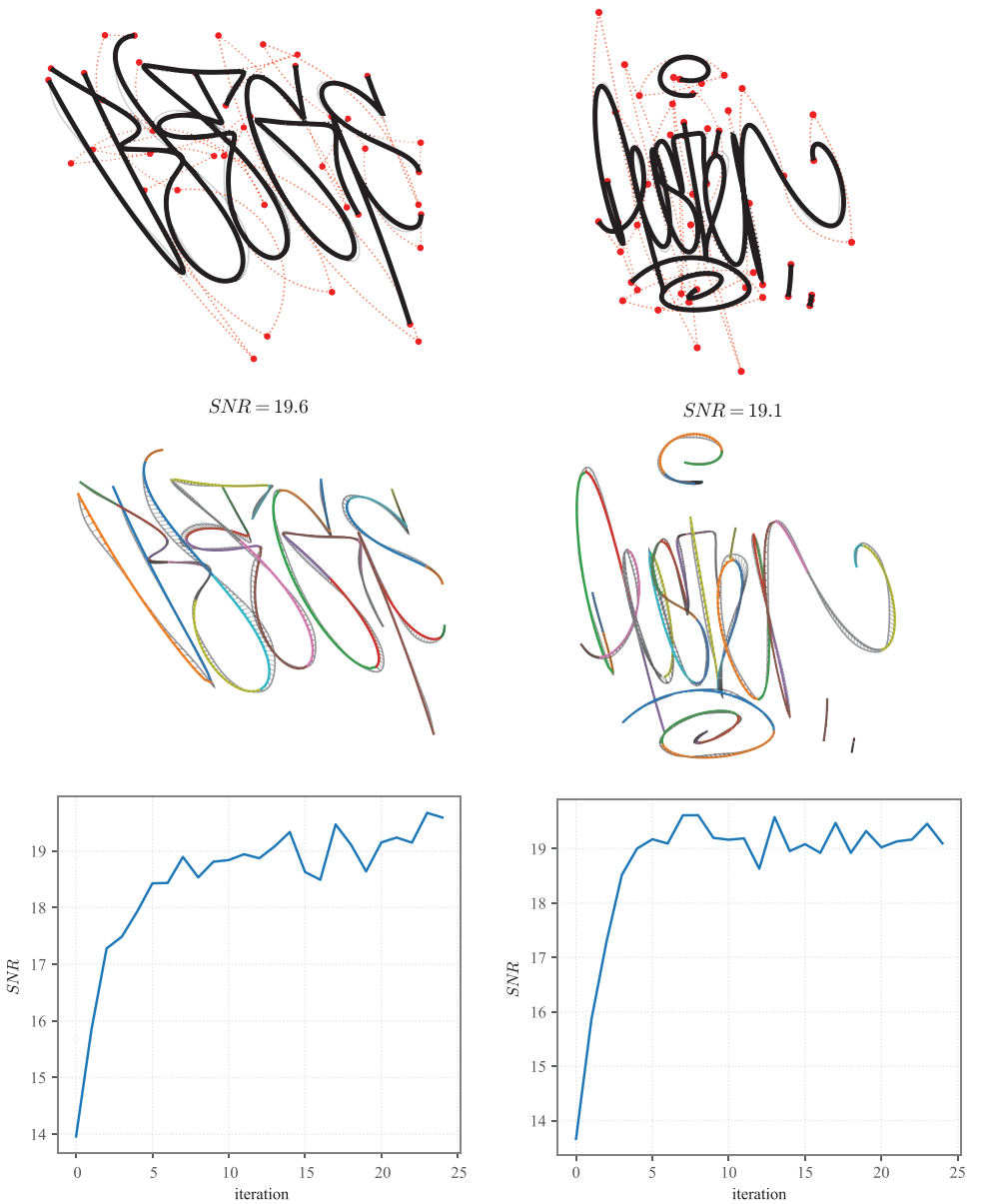


Figure 8.7: Additional examples of graffiti tag reconstructions (with data from the Graffiti Analysis database (Roth et al., 2009)) together with the corresponding SNR plots.

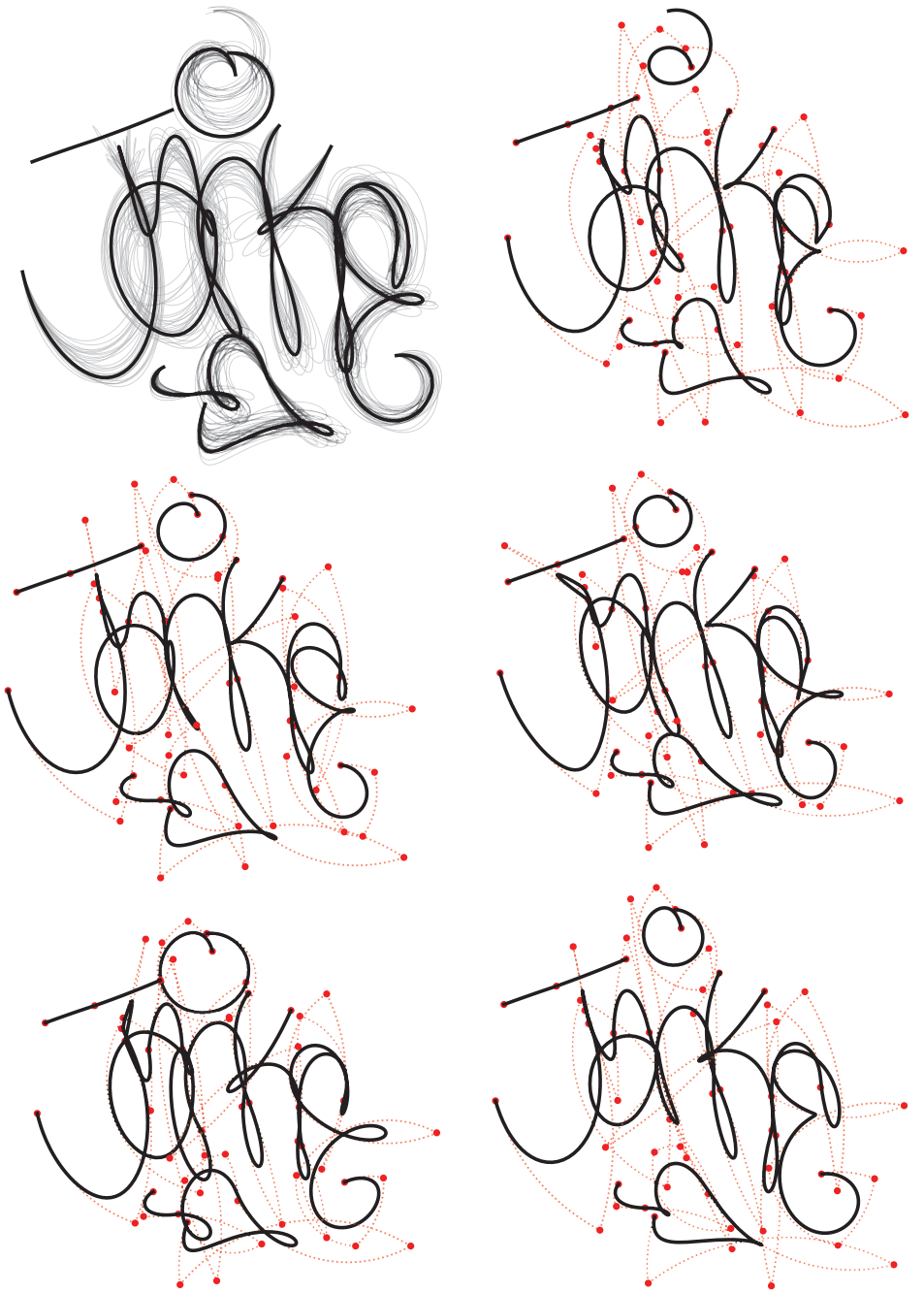


Figure 8.8: Parametric variations of a reconstructed graffiti instance from the Graffiti Analysis database (Roth et al., 2009). Top left, the original reconstruction (black trace) overlaid with 30 variations. Note that variability is higher in proximity of smooth segments of the trajectory. The remaining traces are randomly perturbed samples, with the corresponding (perturbed) action-plan in red.

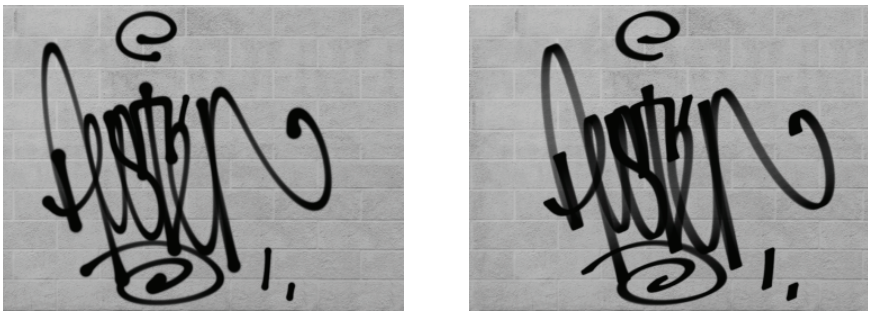


Figure 8.9: Example of content generation. Tags reconstructed from the GML (Graffiti Markup Language (Roth et al., 2009)) format, and rendered with kinematics based brushes over a wall texture. Original on the left, reconstructed version on the right.

In our case, we instead rely on the properties of the ΣΛ model, which ensures the resulting reconstruction is smooth and infinitely differentiable (C^∞).

8.3.1.1 Combining variations and iterative refinement

As noted in Chapter 4, simply scaling the Δt_i parameters (Figure 8.10.a) can quickly result in a loss of structure and legibility. Another method to mitigate this effect is to run a second step of the iterative refinement procedure with a lower value of $\Delta t_-, \Delta t_+$. As a result, we can achieve a smoothing effect while still preserving the structural similarity of the input, as provided by the original features. Variable degrees of smoothing can then be achieved by interpolating the ΣΛ parameters between the original reconstruction and the smoothed one with a parameter $\alpha \in [0,1]$ (Figure 8.10.b).

More flexible stylisation effects can also be achieved with a similar approach, for example by constraining all stroke curvature parameters δ_i to a user-specified value (Figure 8.10.c) and then running the iterative refinement with $\lambda_\delta = 0$, hence not further affecting the parameters. While we use linear interpolation for the parameters $\Delta t_i, \delta i$, we observe that these are not linearly related to the target positions. While this relation deserves further analysis in future studies, we achieve satisfactory results by specifying a power of α for interpolating targets (Figure 8.10.d) and observe experimentally that a power of 7 works particularly well for our use case (Figure 8.10.b & c).

8.4 Comparison: constrained minimum jerk model and MIC

The ΣΛ reconstruction method discussed in this chapter infers the kinematics of a trajectory, which closely approximates the geometry of a given input trace. This procedure is conceptually similar to the path constrained minimum jerk model (Todorov and Jordan, 1998), which produces a minimum jerk trajectory that interpolates a given sequence of passage points. If a sufficient number of points are selected along an input trace, the resulting trajectory closely

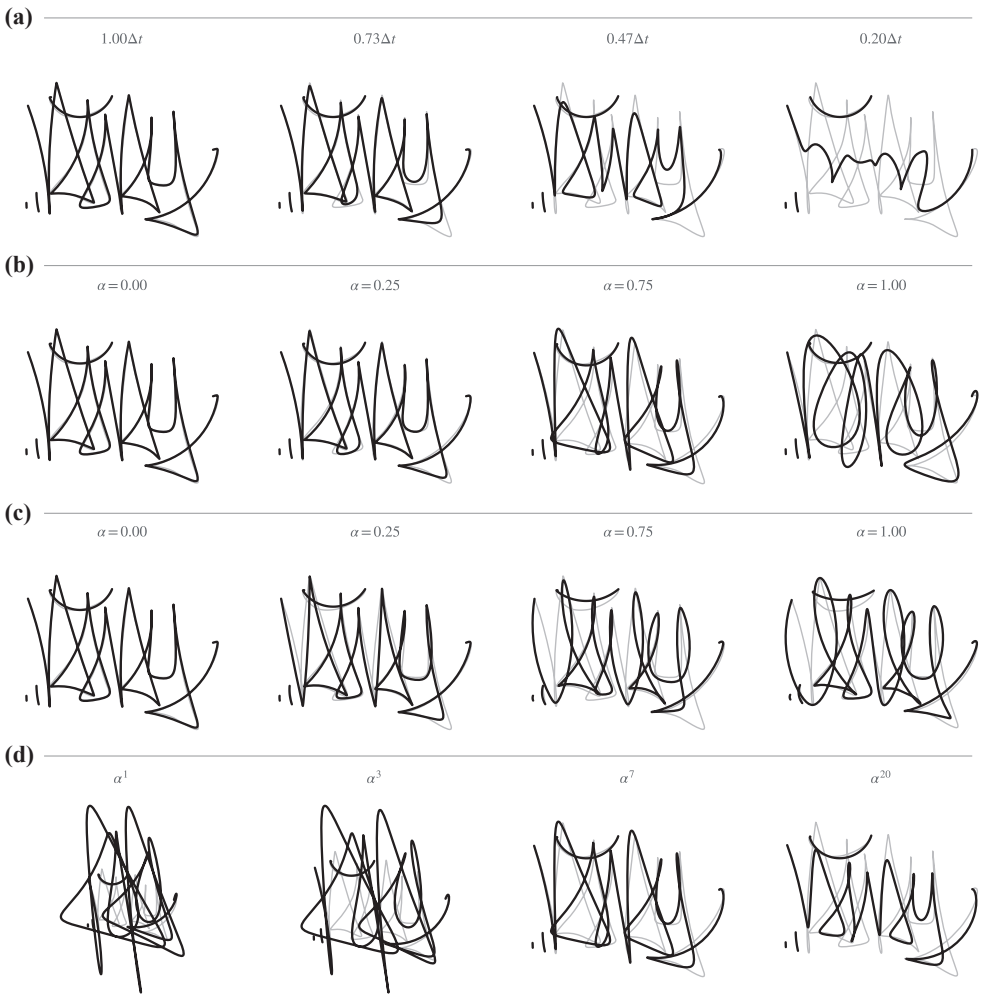


Figure 8.10: Comparison of smoothing and stylisation methods. Row **(a)**, smoothing by global scaling of the time offset parameters Δt_i. Row **(b)**, smoothing by using the parameter α to interpolate between the $\Sigma\Lambda$ parameters of two reconstructions with different values for Δt_+. Row **(c)**, stylisation effects achieved by interpolating between the $\Sigma\Lambda$ parameters of two reconstructions, where the second is performed with user specified values of δ_i. Row **(d)**, effect of different powers of $\alpha = 0.75$ used to interpolate the virtual target positions between the two reconstructions used in row **(b)**; The examples in row **(b)** and **(c)** use a power of 7.

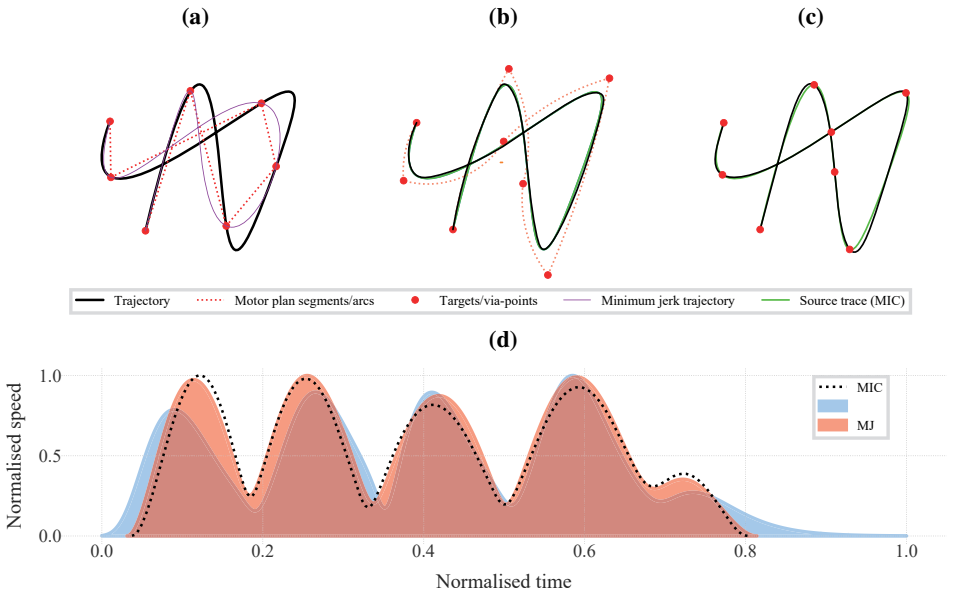


Figure 8.11: Comparison of ΣΛ and path-constrained MJ reconstructions of a trajectory generated with MIC. **(a)** MIC trajectory (black) generated for the motor plan in red. The motor plan was specifically chosen to emphasise the parameterisation problem. The trajectory in magenta is the one that would be predicted by the minimum jerk model for the same motor plan. **(b)** ΣΛ reconstruction (black) of the MIC trajectory (green) and the resulting motor plan. Here the motor plan is visualised with vertices connected by circular arc segments. **(c)** Minimum jerk reconstruction (black) of the MIC trajectory (green). The red points are the locations of CSF extrema and inflections computed along the MIC trajectory during the ΣΛ reconstruction procedure. **(d)** Comparison of the speed profiles for the ΣΛ (blue) and MJ (red) reconstructions together with MIC (dashed-black). The speed profiles are normalised and aligned in order to facilitate visual inspection.

approximates that trace. When the passage points correspond to the vertices of a motor plan, the same method produces a minimum jerk trajectory according to the original formulation of Flash and Hogan (1985).

In Chapter 5 we have seen that a third order interpolating MIC trajectory obeys a jerk-minimising cost function, but the resulting trajectory differs from the one predicted by the minimum jerk model. This difference is due to the uniform parameterisation assumption made in MIC, which, similarly to splines (Lee, 1989), results in a trajectory that does not necessarily interpolate the motor plan at curvature extrema. This can be considered a limitation, because the converse is a property that is especially desirable for user interaction with interpolating curves (Yan et al., 2017).

The possibility to reconstruct ΣΛ parameters from geometry allows us to compare these three models, in order to gain more insights on their relationships and to inform possible avenues of future work. The top row of Figure 8.11 shows a third order interpolating MIC trajectory and its reconstruction with the method described in this chapter (Fig. 8.11b) and with the path-constrained minimum jerk model (Fig. 8.11c). For the MIC trajectory in Figure 8.11a, we specifically choose a motor plan that emphasises the parameterisation problem and the difference between the trajectory produced by MIC (black) and the one produced by the minimum jerk model (magenta). The ΣΛ reconstruction is done with a default time overlap between lognormals of $\Delta t_i = 0.2$ and fixed μ_i and σ_i, given by intermediate parameters $A_{ci} = 0.1$ amd $T_i = 0.3$. The reconstruction results in a different motor plan from the one used to produce the MIC trajectory in Figure 8.11a. The constrained minimum jerk reconstruction in Figure 8.11c is done by choosing a series of passage points along the MIC trajectory, corresponding to the CSF extrema computed during the ΣΛ reconstruction procedure, together with the inflections (two in this case).

Both the ΣΛ and constrained minimum jerk methods produce a close approximation of the MIC trace. However, it is especially interesting to visually compare the speed profiles produced by the three methods (Fig. 8.11d). To perform the comparison, we normalize the speed profiles and align them by first computing the location of speed minima and then shifting and scaling the profiles horizontally, in order to minimise the distance between minima in a least squares sense. This reveals that the MIC speed profile and the minimum jerk one are nearly identical. Indeed, the MIC formulation in the example produces a minimum jerk trajectory, but not the one predicted by the minimum jerk model (Flash and Hogan, 1985) for the motor plan given in Figure 8.11a.

By construction, the ΣΛ model produces speed profiles consisting of asymmetric bell shapes, which differ from the symmetric ones predicted by the minimum jerk model (c.f. Figure 3.1a, Chapter 3). However, the relative spacing between speed minima in the ΣΛ and minimum jerk remains very similar. This suggests that the procedure used by the ΣΛ model to produce smoother trajectories, i.e. increasing the time overlap between adjacent lognormals, results in an *approximate* isochrony that is similar to the one that is predicted by the minimum jerk model through an optimisation procedure. This relation is worthy of further investigation, especially in sight of finding efficient solutions to the parameterisation problem that affects MIC similarly to splines. Furthermore, the high derivatives of the ΣΛ model can be computed efficiently and at significant locations such as key points (corresponding to curvature extrema). This suggests that it should be possible to implement a similarly efficient constrained optimisation procedure that adjusts the ΣΛ parameters in order to minimise the square magnitude of jerk or some other higher derivatives of position.

8.5 Conclusions

In this chapter, we developed and explained a systematic method to reconstruct ΣΛ parameters from solely a static geometric trace (left by handwriting or drawing gestures). The method is capable of producing an accurate geometric reconstruction of the input, while inferring plausible kinematics underlying its generation on the basis of just an ordered sequence of points as an input. We achieved our goal of a plausible reconstruction of the kinematics by designing a method exploiting a notion of Curvilinear Shape Features (or CSF) to incrementally adjust the temporal and spatial parameters of the ΣΛ model. The method consistently produces accurate (> 15 dB SNR) reconstructions of the input, while providing flexibility for the use of additional constraints that can be exploited in order to generate interactive stylisations and variations. In the following chapter we will see how the same reconstruction can be used to implement an example-driven stylisation procedure.

The reconstruction method relies on an ad-hoc procedure that iteratively reduces the error between the input trace and the reconstructed one. In future work, it would be interesting to exploit the availability of higher order derivatives of the ΣΛ model in order to develop solutions using constrained optimisation methods with stronger convergence guarantees. Another interesting future line of work is to develop more sophisticated methods of setting the ΣΛ parameters μ_i and σ_i, which are currently tuned manually and kept constant. This shall require to explore in depth how the inferred kinematics relate to human data and are perceived by human observers.

We started an experimental evaluation of the ΣΛ model in the context of an ongoing series of experiments conducted by the Department of Psychology at Goldsmiths. We used a procedure similar to the one described in Section 8.4 to compare the ΣΛ and minimum jerk models to a uniform speed parameterisation of the same trajectory. We then asked participants to rate animations of movements generated with the three models, expecting that the movements generated with the ΣΛ and minimum jerk model would be perceived as more natural and aesthetically pleasing then the movements generated with the uniform model. The experimental results are still under investigation and a manuscript is being prepared with details and analysis at the time of the writing of this thesis. In summary, the participants found the biologically feasible movements (ΣΛ and minimum jerk) more aesthetically pleasing and natural than the uniform ones. In addition, naturalness and aesthetic ratings were positively correlated, suggesting that movement naturalness is a predictor of how beautiful the movement appears. While the results showed no significant difference between the aesthetic and naturalness rating of the ΣΛ and minimum jerk model, the difference was also non-significant between the ΣΛ and uniform model. This makes the results difficult to interpret, and emphasises the need for further investigation on the choice of fixed parameters when performing the ΣΛ reconstruction.

Examining Figure 8.11d shows that the speed profiles produced by the ΣΛ and minimum jerk model are very similar. However, with the given selection of μ_i, σ_i parameters, the ΣΛ reconstruction procedure gives a slower movement at the beginning and end of the trajectory. This emphasises an intrinsic difference between the minimum jerk and ΣΛ model, where the former model makes predictions on the global kinematic regularities of a movement (i.e. the minimisation of jerk), while the latter makes predictions at the planning level and on how kinematics emerge through the superposition of submovements. The minimum jerk model predicts a unique optimal trajectory given a series of passage points, while the ΣΛ model can produce many different kinematic realisations given the same motor plan. As we have seen in Chapter 4, and as we shall see in the next chapter, this property becomes advantageous when generating variations and stylisations of a trajectory. However, this also requires making parameter choices when reconstructing a trajectory from a trace (such as the constant values of μ_i, σ_i or the value of Δt_-), which ultimately impact the inferred kinematics and can influence the perceived naturalness of the resulting movement.

Chapter 9

Example-driven stylisation with the Sigma Lognormal Model

This chapter is based on a collaboration between myself, Prof. Frederic Fol Leymarie, Memo Akten, Prof. Réjean Plamondon and Prof. Mick Grierson. The neural network architecture and code has been principally designed and developed by Memo Akten. Initial results were presented at the MOCO conference in London in June 2017 (Berio et al., 2017a). Section 9.3.1 presents initial results of ongoing work that stems from the useful advice of Prof. Luca Citi at University of Essex.

In the previous chapter, we have seen how CSFs allow us to separate a structural and kinematic component from a trace in terms of a motor plan and the corresponding $\Sigma\Lambda$ parameters. This separation enables the formulation of an example-based stylisation procedure that is conceptually similar to ones that have been proposed for the tasks of stylising vector paths (Hertzmann et al., 2002; Lang and Alexa, 2015) and images (Hertzmann et al., 2001; Gatys et al., 2015). Such methods consider stylisation as a bi-level transfer problem commonly referred to as "style transfer": given two input patterns (paths, images), generate a third one with the structure or "content" of the first but with a style that is visually similar to the second. The implementation of such a method clearly requires a method that can distinguish from a given input pattern, a descriptor of style and one of structure.

In the proposed framework of "style as kinematics" the inputs to the system are the traces of instances of writing, calligraphy or graffiti. We represent (i) "structure" with a motor plan, and (ii) "style" with a set of kinematic parameters that determine the fine evolution of movements that follow the same motor plan. The kinematic parameters are expressed with the $\Sigma\Lambda$ model, and the previously described reconstruction procedure allows us to recover these parameters and a motor plan from one or more input traces. To implement an

example-driven stylisation method, we first learn a mapping between the motor plan (structure) and the $\Sigma\Lambda$ parameters (kinematics) for one or more given inputs and use this mapping to predict the parameters for another motor plan, which may be provided by a user or automatically extracted from another trace. The underlying hypothesis is that (i) the parameterisation of the $\Sigma\Lambda$ model is sufficiently rich to capture features of a hand-style that can be transferred between different motor plans, and (ii) that since glyphs and written traces usually consists of repetitive patterns, it should be possible to learn the mapping from a very small number of training examples.

As a proof of concept for the implementation of this procedure, we learn the mapping with a recurrent neural network (RNN), and more specifically a *recurrent mixture density network (RMDN)* (Graves, 2013) with a *long short-term memory (LSTM)* architecture (Hochreiter and Schmidhuber, 1997). Recent developments have shown that these RNNs are capable of modelling complex sequential (or time-ordered) data such as text (Sutskever, 2013), images (Gregor et al., 2015), dance (Crnkovic-Friis and Crnkovic-Friis, 2016) and handwriting (Graves, 2013). While many existing deep learning approaches aim to minimise the use of "hand-crafted features", we hypothesise that for our task it is beneficial to formulate a mid-level mapping that exploits our knowledge of the specific problem domain. Our rationale is that human handwriting (and related artistic processes) results from the orchestration of a large number of motor and neural subsystems, and that movement is arguably planned using some form of higher level mapping, possibly in the form of movement primitives which are then combined in a syntactic manner similar to language (Richardson and Flash, 2002; Flash and Hochner, 2005). For this particular study, we represent this mapping as $\Sigma\Lambda$ movement primitives, which abstracts the complex task of trajectory formation from the neural network, which is then left with focusing on the higher level task of movement planning.

We demonstrate this approach on digitised traces, which are recorded by a user with a pen-tablet or downloaded from a large online graffiti motion database (Roth et al., 2009). As a preprocessing step, we first reconstruct the traces, and then operate on the inferred motor plans and $\Sigma\Lambda$ parameters. Compared to point-sequences, this representation is more *concise* (i.e. with low cardinality) and *meaningful*, such that every representative locus is now a high level segment of the trajectory. Such a representation is also *resolution independent*, and can easily be manipulated prior to and after training. In addition, the $\Sigma\Lambda$ parameterisation can be exploited to augment the training data for the RNN with new samples that mimic the variability one might observe when an artist draws the same form multiple times. We exploit this capability to augment our training data and learn from very small amounts of original input datasets — as small as a single training example.

9.1 Method

The ΣΛ reconstruction procedure recovers a motor plan P and a set of kinematic parameters Θ_P which closely approximate a set of input traces Z with a kinematic realisation $\mathscr{P}$. This allows us to construct an example-based stylisation procedure with a relation similar to the one defined by Hertzmann et al. (2002) for curves:

$$P : \mathscr{P} :: Q : \mathscr{Q} \quad , \tag{9.1}$$

i.e. an example motor plan P is to its kinematic realisation $\mathscr{P}$ as a second motor plan Q is to another kinematic realisation $\mathscr{Q}$. Given P, $\mathscr{P}$ and Q we seek to generate $\mathscr{Q}$.
Assuming that $\mathscr{P}$ and $\mathscr{Q}$ are generated with (ΣΛ) kinematic parameters Θ_P and Θ_Q the relation becomes

$$P : (P \odot \Theta_P) :: Q : \left(Q \odot \Theta_Q\right)$$

where P and Θ_P are reconstructed from Z, Q is given, and the ΣΛ parameters Θ_Q are unknown and must be inferred so that they visually satisfy relation 9.1. To infer Θ_Q, we train a model that learns a mapping f between P and Θ_P, so that $f(P) \approx \Theta_P$ and the unknown parameters are given by $f(Q) = \Theta_Q$.

9.1.1 Example-based input

The input to the system consists of one or more examples recorded with a digitiser device, such as a tablet or mouse, where each example consists of a set of discrete traces Z. As a first step, we preprocess this input data and reconstruct the traces in the form of ΣΛ primitives, so each example is approximated with a motor plan P and a set of kinematic parameters Θ_P. We use this intermediate representation to train an RMDN model that learns to predict the ΣΛ parameters corresponding to a given motor plan. We call this procedure *kinematic parameter prediction* or KPP (Section 9.1.4). In order to train on small datasets, we augment the preprocessed data by introducing artificial variability at the ΣΛ parameter level. In the following sections we describe the steps that constitute our system and then demonstrate its use with a number of examples in Section 9.2.

9.1.2 Kinematic parameters

Each motor plan P is represented as an initial position $\boldsymbol{p}_0$ followed by a sequence of virtual targets $\boldsymbol{p}_1, \cdots, \boldsymbol{p}_M$. Each pair $(\boldsymbol{p}_{i-1}, \boldsymbol{p}_i)$ corresponds to a ΣΛ sub-movement aimed at $\boldsymbol{p}_i$ and characterised by a pair of *kinematic parameters* $(\Delta t_i, \delta_i)$. When a motor plan consists of multiple paths, we can either (i) consider each path as a different motor plan, or (ii) concatenate all paths into one motor plan, so that the segments between the end of one path and the beginning of the next correspond to "in the air" parts of the movement where the writing tool does not touch the surface. When this is the case, we set the kinematic parameters Δ_i and δ_i

to predefined values (0.5 and 0 in the examples given). Doing so results in a model that also takes into account the order in which strokes are drawn, and, as we will see next, this choice impacts the predictions made by the model. The remaining $\Sigma\Lambda$ parameters A_{c_i} and T_i (or accordingly μ_i, σ_i) are fixed to the predefined values chosen during the reconstruction step, with the same underlying assumption that these parameters represent properties specific to a writer, and given that their effect on the shape of a trajectory is negligible.

9.1.3 Data augmentation

So far, we have seen how the the $\Sigma\Lambda$ parameterisation can be exploited to produce realistic variations of a trajectory for stylisation and procedural generation purposes. Now, we exploit this property to generate an *augmented training set*, which allows the system to be trained on very few original training samples. A similar approach has been used for pattern recognition purposes in other studies and applications using the $\Sigma\Lambda$ model (Fischer et al., 2014; Diaz-Cabrera et al., 2018; Leiva et al., 2017).

Given a dataset of N training samples, we use the same procedure described in Chapter 4 to randomly perturb the target positions and parameters of each sample N_p times. This results in a new augmented dataset of size $N + N \times N_p$ where legibility and trajectory smoothness is maintained across samples. Note that this would not be possible on the original input dataset alone, as perturbations for each data point would eventually result in a noisy unrecognisable trajectory. We then also apply random similarity transforms (rotation and uniform-scaling) to the samples, which we observe is beneficial in improving the robustness to variations of scale and rotation in the inputs.

9.1.4 Kinematic Parameter Prediction (KPP)

Given a motor plan P, we would like to predict the corresponding kinematic parameters Θ_P. We train a model to learn the probability distribution of the kinematic parameters $\{\Delta t_i, \delta_i\}$ for the virtual target $\boldsymbol{p}_i$, conditioned on the virtual targets and kinematic parameters leading up to that target, with:

$$\Pr(\Delta t_i, \delta_i \,|\, \boldsymbol{\Delta p}_{1:i}, u_{1:i}, \Delta t_{(1:i-1)}, \delta_{1:i-1}),$$

where $\boldsymbol{\Delta p}_i \in \mathbb{R}^2$ denotes the relative position displacement between the ith virtual target and the next, $u_i \in \{0,1\}$ denotes the pen-up state (0, pen down, 1, pen up). When desired, the pen up parameter allows us to treat a motor plan consisting of multiple paths as a single sequence.

Note that the distribution is conditioned on the kinematic parameters of the previous sub-movements and virtual targets $(\Delta t_{(1:i-1)}, \delta_{1:i-1}, \boldsymbol{\Delta p}_{1:i-1})$ as well as the current virtual target $\boldsymbol{\Delta p}_i$. Conceptually this represents a writer that knows their next target, is aware of the dynamic history of their movement so far, and wants to know what kinematic parameters to use for the next target. We also considered an alternative model conditioned only on the

virtual targets, independent of the previous kinematic parameters, i.e. $\Pr(\Delta t_i, \delta_i \,|\, \boldsymbol{\Delta p}_{1:i}, u_{1:i})$, which conceptually represents a writer unaware of the dynamic history of their movement. In our preliminary studies we found this model to not perform as well.

9.1.4.1 Model details

We implement our model using *Recurrent Mixture Density Networks (RMDN)* with the LSTM architecture. An MDN (Bishop, 1994) models and predicts the parameters of a Gaussian Mixture Model (GMM), that is a set of means, covariance and mixture weights. An RMDN (Schuster, 2000) is a combination of an RNN with an MDN, and outputs a *unique set of GMM parameters at each time-step,* allowing the probability distribution to change with time as the input sequence develops.

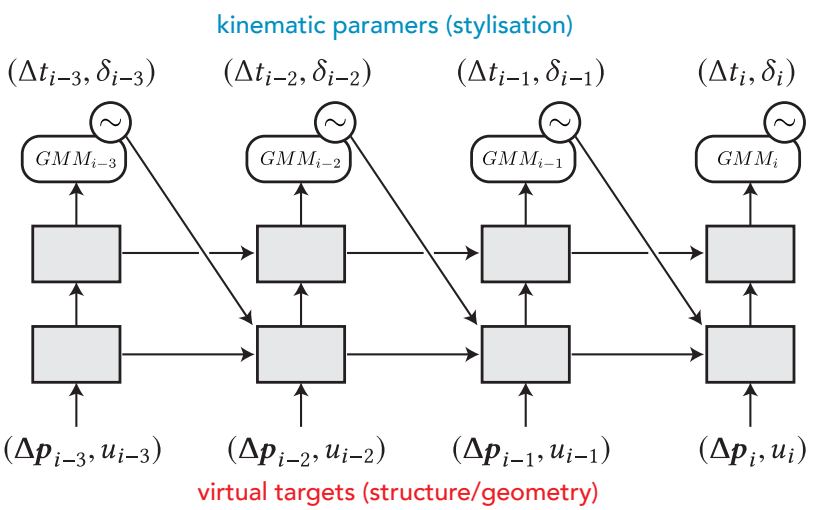


Figure 9.1: Network architecture with two recurrent hidden layers. At each time-step i the network outputs the parameters of a GMM which is sampled (denoted by ~) and fed as input at the next time-step.

The input to the model is given by a vector $\boldsymbol{\xi}_i = [\boldsymbol{\Delta p}_i^\top, u_i, \Delta t_{(i-1)}, \delta_{i-1}] \in \mathbb{R}^5$ where all elements, except for u_i, are normalised to zero mean and unit standard deviation. The desired output is $\boldsymbol{y}_i = [\Delta t_i, \delta_i] \in \mathbb{R}^2$ which consists of the normalised kinematic parameters for the i^{th} sub-movement (Figure 9.1). The probability distribution of the output is expressed as a *bi-variate* GMM with diagonal covariance, which improves training efficiency, since computing the probability distribution function does not require a matrix inversion. This results in a network architecture with an output dimension of $6K$ where K represents the number of components of the GMM. The GMM components are represented as a vector $[\hat{\boldsymbol{\mu}}_i \in \mathbb{R}^{2K}, \hat{\boldsymbol{\sigma}}_i \in \mathbb{R}^{2K}, \hat{\boldsymbol{\rho}}_i \in \mathbb{R}^K, \hat{\boldsymbol{\pi}}_i \in \mathbb{R}^K]$ where the corresponding GMM parameters are given

by (Graves, 2013):

$$\begin{aligned}
&\boldsymbol{\mu}_i^k = \hat{\boldsymbol{\mu}}_i^k : \text{means for } k^{th} \text{ Gaussian, } \boldsymbol{\mu}_i^k \in \mathbb{R}^2 \,, \\
&\boldsymbol{\sigma}_i^k = \exp(\hat{\boldsymbol{\sigma}}_i^k) : \text{standard deviations for } k^{th} \text{ Gaussian , } \boldsymbol{\sigma}_i^k \in \mathbb{R}^2 \,, \\
&\rho_i^k = \tanh(\hat{\rho}_i^k) : \text{correlations for } k^{th} \text{ Gaussian, } \rho_i^k \in (-1,1) \,, \\
&\pi_i^k = \text{softmax}(\hat{\pi}_i^k) : \text{mix weight for } k^{th} \text{ Gaussian , } \sum_k^K \pi_i^k = 1 \,.
\end{aligned} \tag{9.2}$$

At each i^{th} time step, the probability of the kinematic parameters $\boldsymbol{y}_i$, given the input vector $\boldsymbol{x}_i$, is:

$$\Pr(\boldsymbol{y}_i \,|\, \boldsymbol{\xi}_i) = \sum_k^K \pi_i^k \mathcal{N}(\boldsymbol{y}_i \,|\, \boldsymbol{\mu}_i^k, \boldsymbol{\sigma}_i^k, \rho_i^k) \quad . \tag{9.3}$$

9.1.4.2 Training

We use a maximum likelihood objective that minimises the negative log-likelihood (a.k.a. surprisal (Feldman and Singh, 2005)) of generating the input samples. Given a training set of input-target pairs $(\boldsymbol{\xi}_i, \hat{\boldsymbol{y}}_i)$, we define the loss for a single training example with:

$$J_s = -\sum_i^L \ln\left(\Pr\left(\hat{\boldsymbol{y}}_i \,|\, \boldsymbol{\xi}_i\right)\right) \quad , \tag{9.4}$$

where L is the length of the sequence. The total loss is given by summing J_s over all training examples.

We use a form of Truncated Backpropagation Through Time (BPTT) (Sutskever, 2013) whereby we segment long sequences into *overlapping* segments of maximum length L. In this case, long-term dependencies greater than length L are lost, however with enough overlap the network can effectively learn a *sliding window* of L time-steps. We shuffle our training data and reset the internal state after each sequence. We empirically found an overlap factor of 50% to perform well, though further studies are needed to confirm the sensitivity of this choice.

We use *dynamic unrolling* of the RNN, whereby the number of time-steps to unroll to is not set at compile time, in the architecture of the network, but unrolled dynamically while training, allowing variable length sequences. We train using the Adam optimizer (Kingma and Ba, 2014) with the recommended hyper-parameters. To prevent exploding gradients we clip these using the L2 norm as described by Pascanu et al. (2013) and experimentally set the threshold to 5.

We use LSTM networks (Hochreiter and Schmidhuber, 1997) with input, output and forget gates (Gers et al., 2000), and dropout regularization (Pham et al., 2014). We have employed

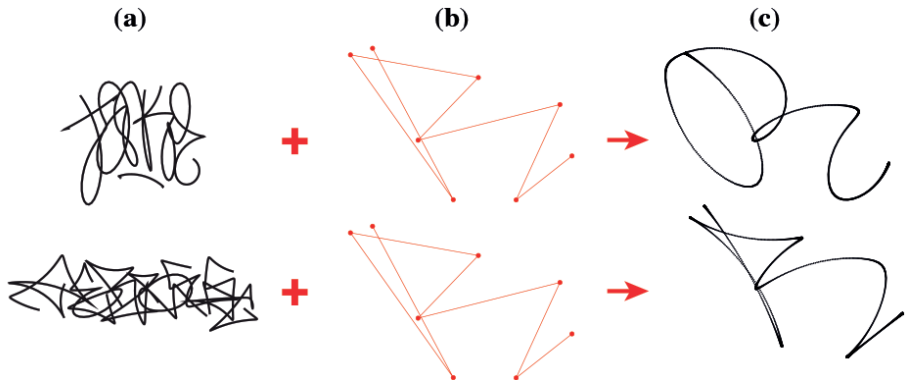


Figure 9.2: Dynamic parameters generated over user specified virtual targets (in red), using *two separate models*, each trained on a single example (with data augmentation x8000). Each row shows: **(a)** the training example, **(b)** the user provided virtual targets and **(c)** the trajectory predicted by the corresponding model. Training examples: top row is from the GML database (Roth et al., 2009), bottom row was drawn using a tablet.

both a grid and a random search (Bergstra and Bengio, 2012) on various hyper-parameters in the ranges: sequence length (5…128), number of hidden recurrent layers (1…3), dimensions per hidden layer (64…1024), number of Gaussians (5…20), dropout keep probability (50%…95%) and peepholes {with, without}. We experimentally settled on an architecture of 2 recurrent layers, 20 Gaussians, dropout keep probability of 90% and no peepholes. The layer and sequence sizes depend on the use cases, which are discussed next.

9.1.4.3 Prediction

To predict the kinematic parameters for an input motor plan, we first scale the input so its size approximately matches the size of the motor plans used for training. Given a qualitative evaluation of the results, we do so by uniformly scaling the input motor plan so that the median distance between consecutive virtual targets is equal in the input and the training set. The prediction is then made by translating and scaling the resulting offsets $\mathbf{\Delta p}_i$ by the same amounts used to transform the training data to zero mean and unit standard deviation. Parameter prediction is executed iteratively, starting from the first offset, stochastically sampling the GMM generated by the network (equation (9.3)) and then feeding back to the network the next offset together with the sampled parameters.

9.2 Results

Given a motor plan, the KPP model (Section 9.1.4) is used to produce different stylisations that resemble the dataset it has been trained on. The input motor plan can be either (i) directly defined by a user or (ii) reconstructed from a set of input traces.

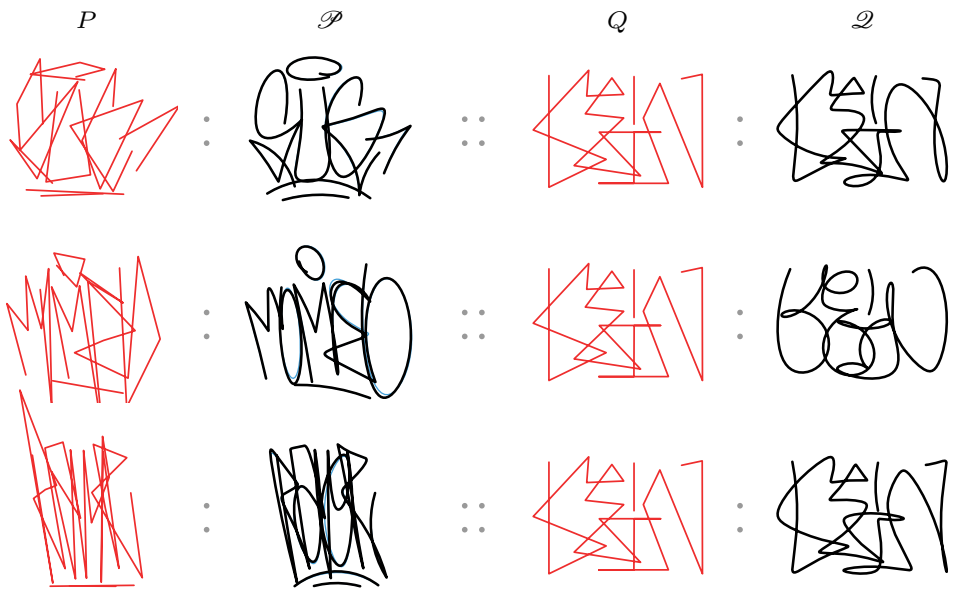


Figure 9.3: Relation (9.1) implemented with KPP models trained on a single example, and with one sequence per stroke. The columns marked with P, $\mathscr{P}$, Q, $\mathscr{Q}$, respectively denote the example motor plan, reconstructed trajectories followed by the user provided motor plan and the stylisation given by the model. Column $\mathscr{P}$ overlays the prediction of the model (black) over the original reconstruction (in cyan) used as training data (the overlaps are nearly perfect). The stylised trajectories are generated with adjusted virtual targets, as described in Section 9.2.1.3.

9.2.1 User defined virtual targets.

In the following application, a user provides an input motor plan and the system predicts various smooth trajectories, the stylisation of which varies according to the kinematic parameters learned from a given training set. The input motor plan consists of a sequence of sparse poly-lines, and it can be specified interactively, or loaded from a vector file. While the training procedure is performed offline, sampling the trained model runs at interactive rates. As a result, the user is able to view the results and interact in real time, for example by dragging the virtual targets around with a mouse, and seeing the final smooth trajectory update instantaneously with desired styles. The training set can contain one or more examples of target styles.

9.2.1.1 Single training examples

As mentioned previously, we can use data augmentation to train a KPP model on as few as a *single* initial training example and the model is able to consistently predict kinematic pa-

Figure 9.4: KPP model trained on a series of multiple strokes. The model predicts order dependent features such as the "wiggle" in the first stroke (grey circles in $\mathscr{Q}$ and $\mathscr{Q}$).

rameters in that style (Figure 9.2). We observe that a maximum sequence length of $L = 15$ works well for this use case. We explore two different methods to specify the training data for the network: with each stroke stored in its own sequence, or with all strokes concatenated in the same sequence (with pen up movements).

One stroke per sequence: We train the network on a set of sequences, where each sequence is corresponds to one (perturbed) stroke from the training example. When using this model to predict kinematic parameters for an input motor plan, the predictions should be made separately for each stroke in the input (Figure 9.3). The resulting model is independent of the order in which strokes are specified, which can be advantageous for example when considering examples and input motor plans for different writing systems. We find that this approach works well with a relatively small network size (64 units per layer) and a data augmentation of $n_p = 3000$, which also results in a relatively fast training procedure when compared to the subsequent methods.

Multiple strokes per sequence: In this case, we train the network on a set of sequences, where each sequence contains all the perturbed strokes from the training example, concatenated in the same order that is specified in the example. This approach also takes into account the order in which the example strokes are specified. For input motor plans that are structurally similar to the training example, this can result in prediction that captures global features, such as the "wiggle" that is being reproduced in Figure 9.4. However, the sensitivity to stroke ordering is high, resulting for example in a completely unrecognizable reconstruction of a training example when the order of the strokes is varied. This suggests a lower generalisation power of this model with respect to the previous one (Figure 9.5). This approach also requires a larger network size (400 units per layer in the given examples) with the same data augmentation of $n_p = 3000$, resulting in a considerably slower training procedure.

9.2.1.2 Priming, multiple examples and variability

When a model is trained with a single stroke per sequence, it is possible to control the stylisation by *priming* the model so it favours predictions for a specific stroke (Graves, 2013).

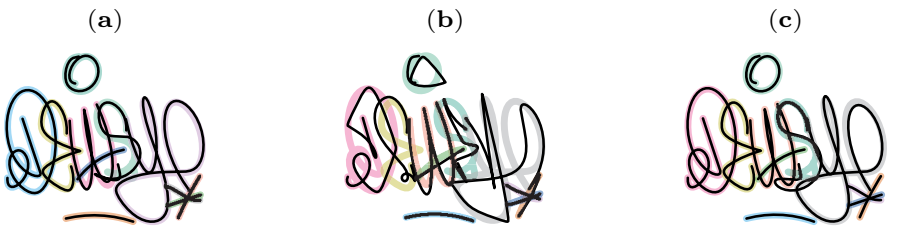


Figure 9.5: Reconstruction of the training example with a KPP model trained with multi-stroke sequences **(a-b)** and one trained on single stroke sequences **(c)**. **(a)** The multi-stroke sequence model correctly reconstructs the training example, when its motor plans are fed in the original order (color coded). **(b)** However, shuffling the order of the input results in a strong degradation of the reconstruction. **(c)** This issue does not affect a model trained on sequences consisting of a single stroke.

Priming is achieved by first feeding the model the sequence corresponding to a stroke, before feeding it the virtual targets which we wish to make a prediction on. This mode of operation can be selected by a user to fine tune the predictions made by the model. For example in Figure 9.6, top row, we force the first strokes of the training and input trajectories to be similar, and thus reproduce the "wiggle" in the training example without the need to train a less efficient order-dependent model.

We also can train a KPP model on a dataset that contains *multiple styles*. In this case we can use *priming* to select which example in the training set determines the stylisation (Figure 9.7). We observe that while a shorter sequence length is sufficient for models trained on a single style, a longer one (e.g. 64) is necessary for this use case, to help it "remember" the primed style across more virtual targets.

Because the model predicts parameters by stochastically sampling a learned distribution, different predictions and consequent variations of the generated trajectory can be achieved by choosing different seeds for the pseudo-random number generator (Figure 9.8).

9.2.1.3 Virtual target adjustment.

By definition, the virtual targets of a $\Sigma\Lambda$ motor plan are imaginary points at which ballistic sub-movements are aimed. Consequently, these points are often not located along the trajectory, especially in correspondence with smooth trajectory portions where two adjacent lognormal primitives have a large degree of overlap (low Δt). When the training examples contain many such smooth portions, the model predictions can result in trajectories that have a degraded structure not sufficiently similar to the one of the input motor plan. This issue can be easily avoided, by following the same iterative key point adjustment procedure described in Section 4.4.2.1. A number of examples in this chapter have been computed with 3 iterations and $\lambda_p = 0.25$.

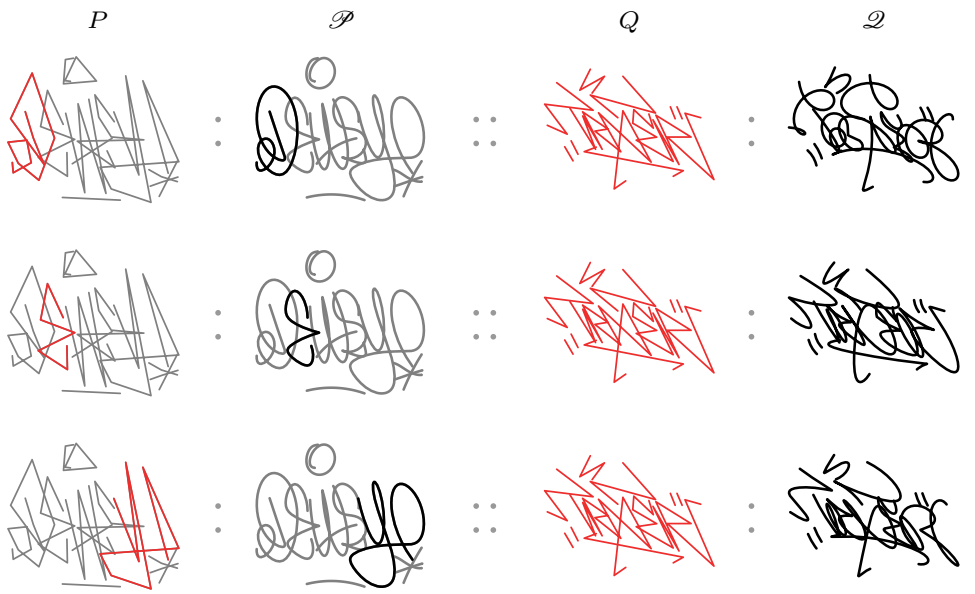


Figure 9.6: KPP model trained on a single stroke sequence and primed on a specific stroke (emphasised in red for the motor plan P and in black for the trajectory $\mathscr{P}$).

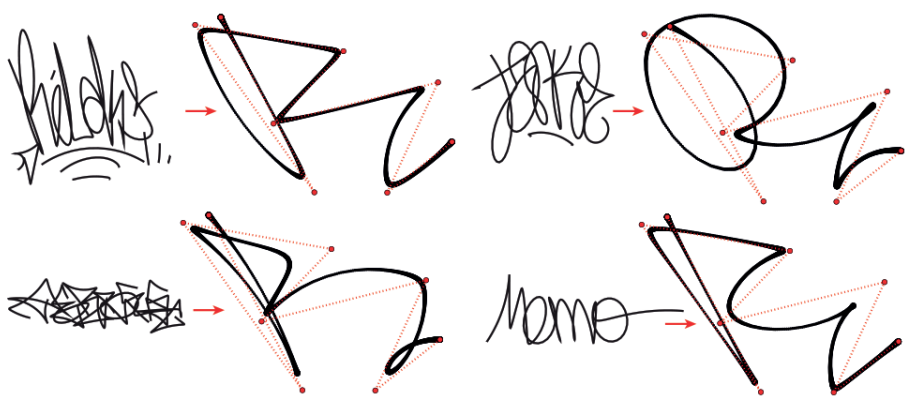


Figure 9.7: Dynamic parameters generated over user specified virtual targets (in red) using *a single model* trained on 4 examples (with data augmentation x2000). Each trajectory has been generated by priming the network with the corresponding example. Training examples: top row are from the GML database (Roth et al., 2009), bottom row were drawn using a tablet.

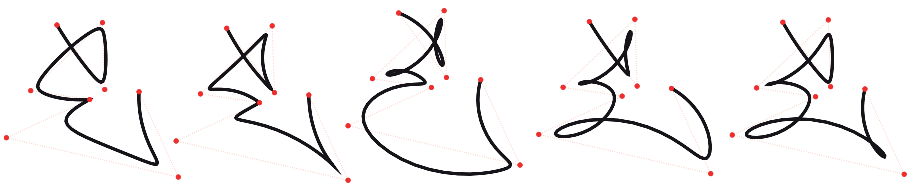


Figure 9.8: Variations generated by varying the random number generator seed prior to sampling a model trained on multiple examples.

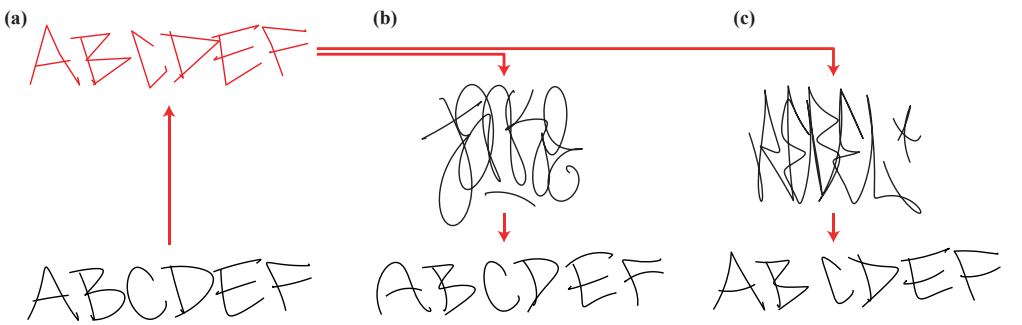


Figure 9.9: Kinematic style transfer of user drawn traces. **(a)** Letters of the alphabet drawn by a user with a tablet (bottom) and corresponding motor plan (top, red). We train two KPP models (**(b)** and **(c)**, top), which predict new kinematic parameters for the motor plan of the original traces and generate the trajectories below.

9.2.2 Kinematic Style Transfer

The same methods described above can be extended to arbitrary input traces, either drawn by a user or taken from an existing online dataset. Given such an input trace, we first use the $\Sigma\Lambda$ parameter reconstruction method to extract a series of virtual targets and corresponding kinematic parameters. We then *discard* the reconstructed kinematic parameters and replace them with the ones predicted for the corresponding virtual targets by a given model, identically to the previously demonstrated examples. The result is a kinematic analogue of style transfer procedures such as the ones described for images by Gatys et al. (2015), which we call "kinematic style mixing". The resulting output is structurally similar to the input trace, but possesses kinematic and geometric features derived from the training examples.

We test the method with a simple sequence of letters drawn by a user with a tablet (Figure 9.9.a, bottom) and observe that, depending on the example used to train, or prime the network, the method produces clearly different and readable stylisations of the input (Figure 9.9.b,c). On the other hand, the quality of the results strongly depends on the structural complexity of the input and on the perceptual similarity of the reconstructed virtual target sequence to the input path. In some more complex examples, the initial readability of the

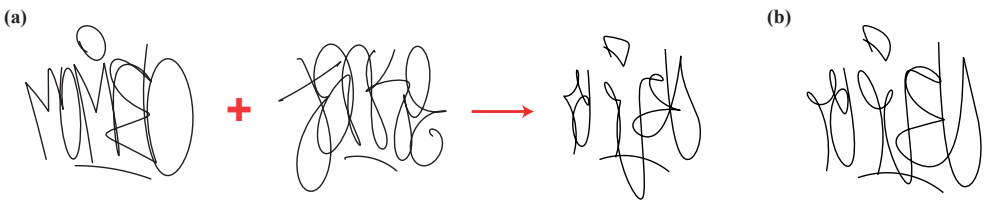


Figure 9.10: Less satisfactory case for the stylisation of a complex tag. **(a)** Left: drawn trace to be stylised, Middle: training example, Right: Output of the model, applying the style of the training example to the user drawn trace. **(b)** Result after iterative adjustment of the virtual targets.

input can be lost and the quality of the resulting stylisation is not satisfactory, or in general difficult to evaluate qualitatively (Figure 9.10.a).

To improve the structure and recognisability of the stylised traces, we can again use an iterative procedure that adjusts the virtual target positions. In this case, we re-run the iterative refinement step used to reconstruct the input, but replacing the kinematic parameters of the original reconstruction with those predicted by the model. We then run the iterative refinement procedure by adjusting only the virtual target positions (with $\lambda_\delta = 0$ and $\lambda_\Delta = 0$) and setting λ_p by a user configurable amount ($\lambda_p = 0.5$ and 3 iterations in Figures 9.10.b and 9.11).

This procedure brings the curvature extrema of the stylised trajectory and the ones of the input closer together, which, based on known results in visual perception (De Winter and Wagemans, 2008a), should improve recognition. Figure 9.11 shows different combinations of inputs (blue column) and examples (red row) that are used to predict new kinematic parameters for the motor plans of the inputs. The examples along the diagonal use the same example for the motor plan and kinematic parameters, which results in a reconstruction of the original trajectories (in blue). While the virtual target adjustment step generally improves readability of the stylised trajectories, the examples in the first row (in black) show that this is not always the case, which emphasises the sensitivity of the method to the structure of the input and suggests that further research is needed to improve the quality of the results.

9.3 Discussion

We developed and tested the system on a commodity laptop. With this configuration, training the model on a single example (64 units, $n_p = 3000$) takes approximately 4 to 5 minutes. More complex models take significantly longer to train. The prediction and adjustment steps run at interactive frame rates.

Figure 9.11: Kinematic style transfer between different examples of tags reconstructed from the GML database (Roth et al., 2009). The blue column shows the input trajectories that are used to recover motor plans, and the red row shows the examples that are used to stylise the recovered motor plans. In black are different stylisations resulting from combinations of the inputs and the examples. The entries along the main diagonal are the predictions of the model for the motor plan it was trained on.

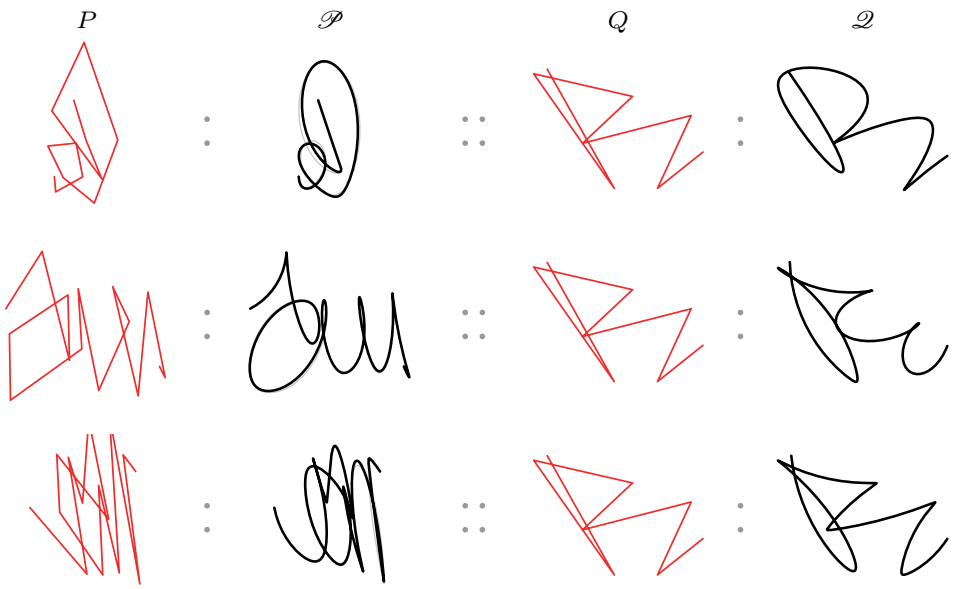


Figure 9.12: Relation (9.1) implemented with VARMA and examples consisting of a single stroke. The column marked with $\mathscr{P}$ displays the reconstruction used as an example (gray) with the trajectory predicted by the model in black. With a single stroke, the prediction is qualitatively accurate.

9.3.1 Model complexity

We have demonstrated that with an appropriate feature representation, it is possible to train a complex and flexible model, such as an RNN, on a very small dataset, with results that we evaluate as qualitatively satisfactory. We have chosen this specific model, given the remarkable results that have been demonstrated by Graves (2013) with handwriting data, and in order to evaluate the performance of this model with the $\Sigma\Lambda$ model as an intermediate representation. At the same time, the sparsity of this representation suggests that similar results can be achieved with simpler methods, possibly resulting in a significant gain in computational performance. A future goal would be to develop a fully interactive system, where a user can quickly stylise a trajectory from examples executed or loaded on the fly.

9.3.1.1 Linear solution

As a step in this direction, we have also tested a simple linear model, perhaps at the opposite end of the complexity spectrum with respect to the RNN: a vector auto-regression-moving average (VARMA) model (Kendall and Ord, 1993), in which the kinematic parameters for one virtual target depend on a window of L previous virtual targets and kinematic parameters.

This can be expressed with the linear quadratic system:

$$\boldsymbol{y}_i = \sum_{k=1}^{L} \boldsymbol{A}_i^k \boldsymbol{y}_{i-k} + \sum_{k=0}^{L} \boldsymbol{B}_i^k \boldsymbol{\xi}_{i-k} + \boldsymbol{C}_i \boldsymbol{\phi}_i + \boldsymbol{\epsilon}_i \quad , \tag{9.5}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ are coefficient matrices, $\boldsymbol{y}_i$ and $\boldsymbol{\xi}_i$ are the kinematic parameters and virtual target displacements for the i^{th} $\Sigma\Lambda$ primitive, normalised to zero mean and unit standard deviation, $\boldsymbol{\phi}_i$ is a non-linear function of $\boldsymbol{x}_i$ and $\boldsymbol{y}_{i-1}$, and $\boldsymbol{\epsilon}_i$ is a (Gaussian) white noise term. The system can be expressed compactly as:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Theta} + \boldsymbol{\epsilon} \quad , \tag{9.6}$$

which is linear in the coefficients $\boldsymbol{\Theta}$ and can be solved with ordinary least squares by setting each row of $\boldsymbol{Y}$ with $\boldsymbol{Y}_i = \boldsymbol{y}_i$, and each row of the matrix $\boldsymbol{X}$ with

$$\boldsymbol{X}_i = \left[1, \boldsymbol{y}_{i-1}, \cdots, \boldsymbol{y}_{i-L}, \boldsymbol{\xi}_i, \cdots, \boldsymbol{\xi}_{i-L}, \boldsymbol{\phi}_i\right]^\top \quad . \tag{9.7}$$

We compute the non linear terms $\boldsymbol{\phi}_i$ with a quadratic function of the current virtual target and the previous kinematic parameters given by:

$$\boldsymbol{\phi}_i = \left[\boldsymbol{\xi}_i^\top \boldsymbol{\xi}_i, \boldsymbol{y}_{i-1}^\top \boldsymbol{y}_{i-1}, \boldsymbol{\xi}_i^\top \boldsymbol{y}_{i-1}\right] \quad . \tag{9.8}$$

We test the ability of the model to reconstruct an increasing number of strokes. With a training example consisting of a single stroke, the model is able to correctly reconstruct the example and to successfully stylise new motor plans (Figure 9.12). However, as the number of strokes in the example increases, the reconstruction quality quickly degrades and the relation between the stylisation of the user provided motor plan and the training example becomes unclear (Figure 9.13).

9.4 Conclusion

We have demonstrated how an RNN-based architecture combined with a physiologically plausible model of human movement, the Sigma Lognormal ($\Sigma\Lambda$), can be used to implement a data driven path stylisation system. Our method functions similarly to existing path stylisation methods used in computer graphics applications (e.g. (Hertzmann et al., 2002; Lang and Alexa, 2015)). However, we propose an approach to stylisation based on the concept of "style as kinematics", in which different styles are given by kinematic variations over a common structure of a hand drawn or written trace. We argue that using a physiologically plausible model of movement as a feature representation then provides a number of advantages with respect to polygonal (point sequence), or spline/interpolation based approaches.

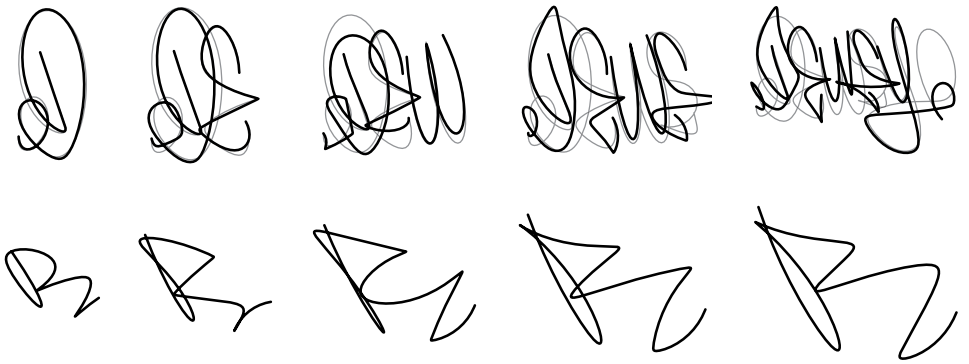


Figure 9.13: Degradation of the parameter predictions with VARMA as the number of example strokes increases (left to right). The first row, is the prediction in black of the example in gray. The second row is the prediction for a user provided motor plan.

First of all, the results reflect a realistic and natural movement which, similarly to the methods developed in the previous chapters, can be used to (i) produce expressive renderings and animations or even to (ii) drive the natural motions of an animated character or robotic drawing device. Furthermore, the $\Sigma\Lambda$ parameterisation provides a concise and informative representation that simplifies the learning task, and its parameters can be used (as demonstrated) to augment training data, but also to generate realistic variations in the generated outputs.

The reported work provides a solid basis for a number of different future research avenues. As an example, we hypothesize that a similar feature representation and architecture can be used to achieve handwriting synthesis results equivalent to the ones demonstrated by Graves (2013), with the additional benefits of resolution independence and the possibility of training on a much reduced dataset size, even achieving satisfying results with one single training example. While the preliminary results achieved with a much simpler model (VARMA) in Section 9.3.1 are not on par with the ones achieved with an RNN, the results still suggest that a slightly more complex data representation, here in terms of the $\Sigma\Lambda$ model, can produce satisfactory results while greatly reducing training time. In the future we also plan to study the performance of other potentially efficient models that can be used to solve a similar problem, ranging from Kalman or particle filtering approaches (Murphy, 2012), to more recent sequence-based deep learning approaches (Zhang et al., 2017b; Tang et al., 2019).

Chapter 10

From 2D Shape to Strokes with CSFs

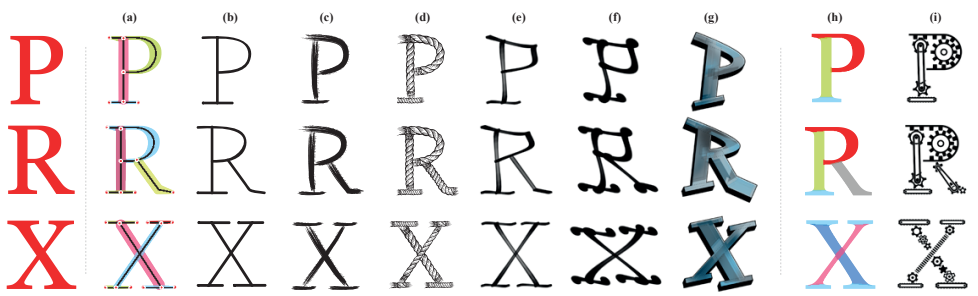


Figure 10.1: In this chapter we partition font outlines (red) into a set of overlapping and intersecting strokes. We recover two stroke representations: (i) a *path* based representation consisting of a set of paths augmented with variable width profiles and annotations describing structural relations among strokes **(a)**; (ii) an *area* based representation that decomposes the input into overlapping shapes **(h)**. In Chapter 11 we will use the first representation to reconstruct a glyph in a variety of stroke-based styles **(b-g)**, from line-based schematizations, to graphic stylizations using skeletal strokes, to artistic stylizations that mimic handwriting and graffiti art. We will use the second representation to compute a similarity metric between stroke areas, allowing consistent shape-based stylizations across the glyphs of a given font.

This chapter and the next one are based on work developed in collaboration with Prof. Frederic Fol Leymarie and with Dr. Paul Asente and Dr. Jose Echevarria at Adobe Research. The collaboration started when I spent a 3 months internship at Adobe Research in San Jose, California, in early 2018. The contents are adapted from a journal article that is in preparation at the time of writing.

Up to this point, we have developed a set of tools that enable the generation of synthetic graffiti with the explicit definition of a motor plan (either user defined or procedurally gener-

ated) or, for the case of tags, by recovering a motor plan from example traces. However, we are still left with the challenge of generating graffiti stylisations of a given string of text, possibly in arbitrary languages or writing systems and with different styles and glyph structures.

One possible solution to this challenge is to first manually define a set of motor plans, for example one for each letter of the alphabet, and to then concatenate and stylise these with the previously defined stroke generation methods. We have seen an example of this approach in Chapter 5. However, this procedure requires a user to manually define the graph structure of these motor plans, which must also be concatenated with consistent spacing, which is also a challenging problem on its own (Haines et al., 2016). A second approach is to train a model that generates motor plans for one or more glyphs with a sequence based model, such as the one developed by Graves (2013) and used in Chapter 9 for trace stylisation. While this approach is interesting and a promising avenue of future research, it requires training data that may be missing for the language or writing system of choice.

This chapter describes a flexible solution that relies on the outlines of existing font as a source for possible letter structures. Recall that a font consists of multiple glyphs (in a certain "style") and a glyph is an element of a font, usually representing a letter, number, or another symbol. The underlying observation of our method is that the outlines of most glyphs can conceal a latent structure as a set of strokes, which when combined, closely re-generate the glyph's shape. Recovering these strokes transforms the wide variety of available digital fonts into a source of possible glyph structures, which can be used to generate structurally-aware stylisations of the glyphs with the methods developed in the previous chapters. Grounding text stylisation on fonts also has the advantage that it can use embedded kerning information to create appropriate inter-glyph spacing, something that can be difficult to achieve with methods that create stylised text from scratch (Haines et al., 2016).

Some simple glyphs can be segmented into strokes just by analyzing their medial axis branching structure (Wang et al., 2013), but this approach is not sufficiently robust for the variety of shapes and combinations of strokes that occur in fonts. Our proposed solution relies on well-studied principles from visual perception (Wagemans et al., 2011) and it must deal with the same issues raised by the related problem of decomposing 2D object outlines into parts. We have seen in Section 3.8.4 that while this kind of problem also depends on domain knowledge (Spröte et al., 2016), psychophysical results suggest that perceptual grouping (Brooks, 2015) and formulating early part-segmentation hypotheses (Xu and Singh, 2002) are possibly pre-attentive processes that occur very early in the vision process.

We approximate and model these perceptual processes with the aid of CSFs (Chapter 7), which facilitate the definition of a set of measures and representations that fit with a pre-attentive computational model of vision. CSFs, and the resulting CASA, serve as a building block that is used to construct a series of incrementally higher level feature representations. These representations support our assumption of an input generated as a combination

strokes. The overall method is also based on practical considerations aimed at the final goal of enabling stroke stylisations using the methods discussed in Part I of the thesis, this for input glyphs in an "as-wide-as-possible" variety of styles and writing systems.

In a nutshell, we first use CSFs to identify potential pairwise relations between features which we name *splits*; these are located where multiple strokes can potentially cross or overlap. We then constrain the space of possible solutions to the stroke identification problem by defining six types of *junctions*, an intermediate representation that characterizes where and how strokes can intersect or end. Junctions are found iteratively and their identification fully characterizes the recovered stroke structure of the glyph. We then use junctions to produce two related stroke representations. For some stylisations, it is most useful to represent each stroke as a simple outline. For others, we recover a set of *stroke paths* that can easily be transformed into a motor plan and a sequence of stylised strokes.

The proposed method produces structurally and visually plausible stroke-based representations of glyphs, using shape analysis alone (Figures 10.1 and 10.2). While this can produce segmentations that are somewhat different from the traditional structure of the glyph, or from ground truth if it exists, it has the considerable advantage of being agnostic to the symbols used and works with glyphs that do not match any standard structure for a letter. The result is a system that can be applied to most glyphs and languages, and even to other 2D shapes that can be closely approximated by a series of strokes.

10.1 Overview

The input to our method is a set of contours, defining the outline of a 2D shape, possibly with one or more holes. The method works on shapes that can be closely approximated by the union of strokes that can cross and that can have one stroke overlapping another at its end. This includes most, but not all, glyph shapes (Figure 10.2), as well as some other shapes that can be drawn using distinct strokes (Figure 10.23).

Figure 10.3 summarises the main steps of the proposed approach. We infer the stroke structure of an input glyph using a simultaneous analysis of its outline and its internal and external (extended) symmetry axes. This joint representation facilitates a robust and accurate estimation of outline features, such as tangents at concavities, and enable a perceptually inspired measure of good continuation along disjoint outline segments. Pairs of concave CSFs are associated with line segments that are called *splits* (Section 10.3), delimiting regions where two strokes can potentially intersect or where one part of the shape protrudes from another.

We furthermore analyze CSFs, splits, and their relationships to the shape's interior skeleton to create higher-level features that we call *junctions* (Section 10.4), describing topological relations between strokes and morphological shape features. We propose six types of junctions, each implying grouping operations on a subset of the SAT and segmenting it into

Figure 10.2: The targets for our method are glyphs that have a recoverable stroke structure, such as the first three glyphs from the left (each a sample from the Rockwell, Giddyup and Apollo fonts), but not the glyph on the right (from the Rosewood font). The inferred stroke reconstruction can be exact (Rockwell, Giddyup) or deviate slightly from the glyph's outline (Apollo). Our method works with glyphs having nonstandard structures, like Giddyup and Apollo, which would present challenges for template-based approaches. In the second row are stroke-based stylisations, produced by our system, of the first three glyphs.

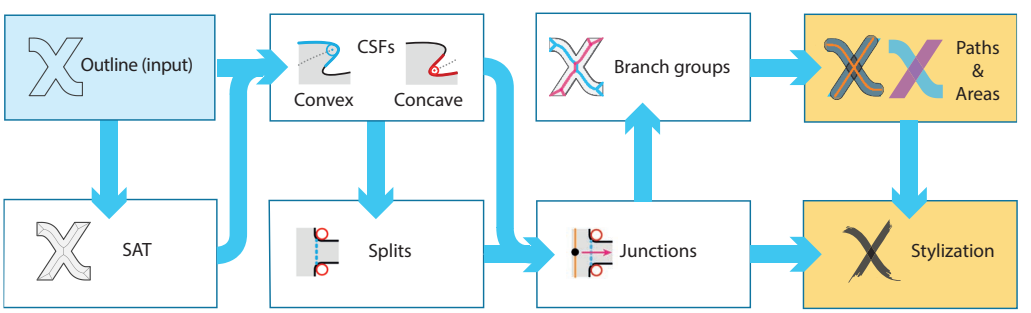


Figure 10.3: High level overview of the segmentation and stylisation of a glyph outline, where the blue arrows show mappings between key elements of our approach. We note that junctions abstract combinations of splits and concavities and determine their relationships to the SAT, ultimately leading to its segmentation and to the construction of stroke representations that can be used to regenerate, animate and stylise the input.

branch groups, potentially overlapping subgraphs that correspond to strokes. This segmentation is transformed into two stroke representations (Section 10.5), which we will use in the next chapter (11) to create gylph stylisations and animations .

10.2 2D Shape Analysis

Perceptual studies show that 2D shape understanding in humans is driven by a combination of cues obtained from an analysis of the boundary, the interior structure including local symmetries, and global properties like the relationships among parts (De Winter and Wagemans, 2006). In accordance with these results, we use a mixed contour-based and region-based approach relying on CSFs and on the interior and exterior CASA, SA^I_+ and SA^E_+, presented in Chapter 7. This extended structure relates the topological structure of the interior to features along its outline and serves as a basis for the subsequent identification of higher-level features that guide the decomposition of the input into strokes.

10.2.1 Extended 2D Shape Analysis

We use CSFs and the CASA to derive a number of additional features and measures that will be useful in the subsequent segmentation stage (Section 10.3).

10.2.1.1 Concave features: Tangents, bisectors, and influence

Each concave CSF is assigned a pair of unit tangent vectors $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ at the first and last points $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ of the corresponding contact region and a unit bisector $\boldsymbol{b}$ with direction $\boldsymbol{t}_1 + \boldsymbol{t}_2$, positioned at the CSF extremum (Figure 10.4). The bisectors are similar to Leyton's "process arrows" for absolute minima of curvature, and the flipped bisector direction captures the hypothetical direction of a force that, when locally applied to a somewhat plastic or malleable version of the outline, creates an indentation (Leyton, 1988).

We use these bisectors to quantify the *influence* of a concavity c on a given SA^I_+ vertex v. The influence depends on the angle ϕ between the bisector and the vector connecting v to the extremum of c. It is given by a Von Mises-like function (Feldman and Singh, 2005), a circular analog of a Gaussian:

$$d(c,v) = e^{k_c(\cos(\phi)-1)} \quad . \tag{10.1}$$

The influence value has its maximum value of 1 when $\phi = 0$ and tends towards a low positive value ($d(c,v) = e^{-2k_c}$) as ϕ increases. As an example, in Figure 10.4.b, the concavity on the left has less influence on the fork (black dot) than the one on the right. The constant k_c determines the minimum influence of a concavity on a vertex and is experimentally set to $k_c = 0.5$.

10.2.1.2 Concave features: Ligatures

SA^I_+ vertices with ribs terminating in the contact region of a concave feature identify *ligatures* (August et al., 1999), thus relating SA^I_+ segments to concavities. Ligature segments can act as "glue" that connects perceptually distinct outline parts (Macrini et al., 2008) and often appear distorted with respect to the ideal spine of a stroke that traverses the corresponding glyph region. To identify ligatures, we first assign each degree-2 vertex in SA^I_+ to its nearest fork or terminal using a radius-weighted distance:

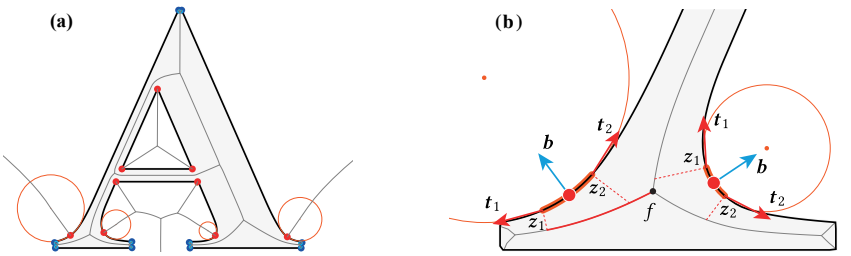


Figure 10.4: **(a)** A capital letter "A", with its interior and exterior medial axes and elements of its CSFs. **(b)** Close up of the bottom-left part, showing ribs connecting contact regions to SA^I (dashed red segments), concave feature tangents (red arrows) and bisectors (blue arrows). The ligature segment produced by the left concavity with respect to the fork (black circle) is emphasized in red.

Definition 10.2.1. The *radius-weighted distance* between a fork or terminal and a degree-2 vertex is $s-r$, where r is the radius of the fork or terminal disk and s the length of the shortest geodesic path through SA^I_+ connecting them.

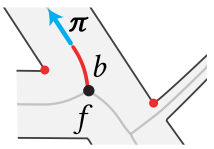
A *ligature* is a contiguous subsegment of a branch b that starts from one of its end-vertices v and extends along adjacent vertices that (i) are closer to v than to the other end-vertex and (ii) have a rib terminating in the contact region of a given concavity c. The segment exists only if any normal (degree-2) branch vertex that is assigned to v has a rib terminating in the contact region of c. A ligature can also be defined with respect to multiple concavities along b, in which case it is the union of all the concavities' ligatures.

10.2.1.3 Assigning concavities to forks

We use ligatures to assign a set of concave CSFs to each SA^I_+ fork. Each concavity c produces zero, one, or two ligatures for each branch in SA^I_+, with one possible ligature for each end-vertex v. If only one ligature exists, and v is a fork, the concavity is assigned to the fork. If two ligatures exist, we select the end vertex for which the influence $d(c,v)$ is highest, or both if the influences are equal, which can happen in symmetric shapes. If any of these vertices is a fork, the concavity is assigned to it. Note that a concavity can be assigned to more than one fork, and that more than one concavity can be assigned to a single fork. In Figure 10.4.b, both concavities are assigned to the central fork with the dot, and no concavities are assigned to the forks near the end of the serifs.

10.2.1.4 Ligatures: Branch protruding direction, $\boldsymbol{\pi}(b,f)$

We also use ligatures to compute the direction of a branch relative to a fork.



Definition 10.2.2. The *protruding direction* $\boldsymbol{\pi}(b,f)$ of a branch b connected to a fork f is given by the first unit tangent vector (blue arrow) along the branch that is not part of a ligature (red branch segment). If the whole branch is a ligature, $\boldsymbol{\pi}(b,f)$ is the tangent at f. A branch connecting two forks in SA^I_+ has two protruding directions, one for each fork.

We will later use $\boldsymbol{\pi}(b,f)$ to relate branches to concavities based on a further analysis of the outline that also depends on the structure provided by CSFs (Section 10.3.2).

10.2.1.5 Branch salience, $\beta(b,f)$.

To distinguish between SA^I_+ branches that characterise the body of a stroke from those that identify morphological features like the cap of a stroke or a corner, we define the salience β of a branch b protruding from a fork f. We take the interior degree-2 vertex of the branch adjacent to the fork and identify the two distinct outline points $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ touched by its disk. β is a function of "stick-out" (Hoffman and Singh, 1997): i.e. the length s of the connecting path along the outline having most ribs incident to b, divided by the length of the connecting straight chord, $\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|$. β is defined using an "exponential-rise-to-maximum" function (Dresp-Langley, 2015):

$$\beta(b,f) = 1 - \exp\left(\lambda\left(1 - \frac{s}{\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|}\right)\right) \quad , \tag{10.2}$$

giving it a range of $[0,1]$. We consider a branch b to be salient with respect to a fork f if $\beta(b,f) \geq 0.5$. If the outline points are on different paths, then $\beta(b,f) = 1$. The parameter λ determines the subdomain for which $\beta \ll 1$ by controlling the steepness of the salience curve. Here we use $\lambda = \lambda_s$, which we empirically set to 0.5. This corresponds to a stick-out threshold of ≈ 2.4. In Figure 10.5, if we use 0.5 as the threshold for determining salience, the upper colored branch is not salient, while the center and lower colored branches are.

Salience with respect to a concavity $\beta_c(b,f)$ We can also compute the salience with respect to a concavity c that is assigned to the fork f, which is denoted as $\beta_c(b,f)$. The computation is identical, except that λ also depends on the angle θ_c between the concavity's tangents. This allows us to adjust the salience value of long branches such as the middle one in Figure 10.5, which helps to identify them as extending into a corner or bend. When a branch is opposite a concave corner, its length and stick-out value increase with a rate that is proportional to $1/\sin(\theta_c/2)$ (Shaked and Bruckstein, 1998). Hence, we define:

$$\lambda = \lambda_s \lambda_\theta, \quad \text{where} \quad \lambda_\theta = \sqrt{2}\sin\left(\frac{\theta_c}{2}\right) \quad , \tag{10.3}$$

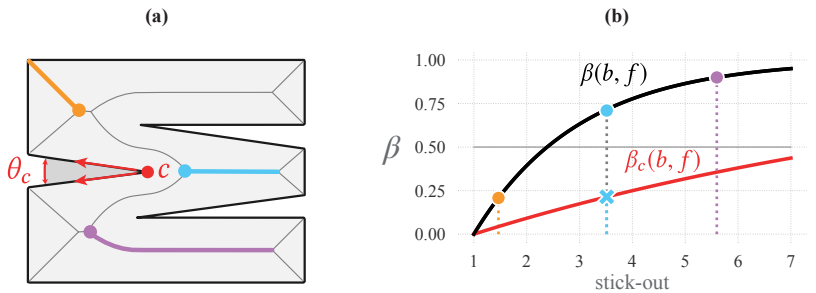


Figure 10.5: Branch salience computation. The salience values $\beta(b, f)$ of the colored branches and originating forks in **(a)** are plotted along the black curve in **(b)**. The blue and pink branches have salience values above the 0.5 threshold. However, the fork for the blue branch has been assigned a concavity c, while the forks for the other two branches have not. This allows computing the salience of the blue branch with respect to that concavity. θ_c is the angle between the tangents of c. Computing the salience with respect to c results in $\lambda_\theta = 0.19$ (eq. (10.3)) producing a flatter salience curve, shown in red. The blue branch is thus considered non-salient because $\beta_c(b, f) < 0.5$ at the blue cross in **(b)**.

which results in $\lambda_\theta < 1$ when θ_c is acute; we note that the stick-out value for which a branch is considered salient increases as θ_c decreases.

10.2.2 Good continuation (α) and flow direction (φ)

Segmenting the input and resolving crossing strokes requires first pairing concavities using a measure of good continuation along the outline. We use *association fields* (Wagemans, 2018), which have been proposed to model the neural processes responsible for contour integration and perceptual grouping in early vision. Various computational implementations have been defined, some based on *cocircularity* (Parent and Zucker, 1989; Yen and Finkel, 1998), i.e. how one local orientation, typically specified by an edge, can be connected to another nearby edge if it is reachable by circular paths within a region specified by the field.

We adapt an experimentally-verified approach by Ernst et al. (2012) that is based on a stochastic model of contour integration (Williams and Thornber, 2001). Given two oriented edge elements, the model defines a field that decays as a Gaussian function of deviation from perfect cocircularity, collinearity, and distance between the two edges (Figure 10.6). We compute the *good-continuation* value α of one concavity with respect to another by selecting two opposing point-tangent pairs $(\boldsymbol{z}_i, \boldsymbol{t}_i)$ and $(\boldsymbol{z}_j, \boldsymbol{t}_j)$; refer to Appendix D.1 for mathematical details of the method. Because the calculation is symmetric, for a given potential pairing, either concavity can be used as the anchor.

For each concavity in a pair there are always two tangents to choose from when computing good continuation. The selection depends on a direction that we call the *flow*.

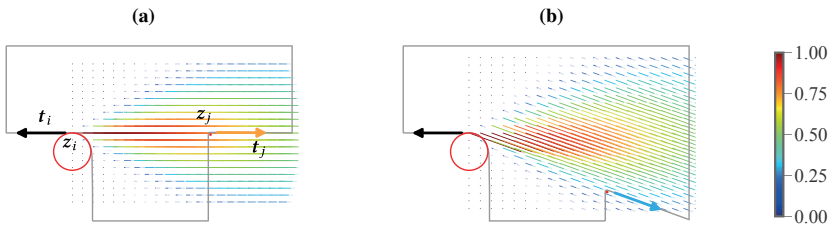
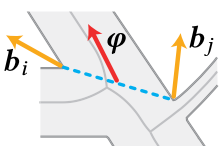


Figure 10.6: Association fields for two corners in a letter T with corresponding colored values α. **(a)** Case where the corners are well-aligned and the association field gives a sufficiently high good-continuation value $\alpha \approx 0.7$. **(b)** Case where the corners are not well-aligned: the association field from one corner reaches the other but only with a low good-continuation value $\alpha \approx 0.2$.



Definition 10.2.3 (Flow, $\boldsymbol{\varphi}$)**.** Given a concavity pair c_i and c_j, the flow direction $\boldsymbol{\varphi}(c_i, c_j)$ is the normalized sum of the two associated bisectors $\boldsymbol{b}_i, \boldsymbol{b}_j$.

When estimating the good-continuation value between two concavities, we always choose the tangent most orthogonal to $\boldsymbol{\varphi}$. We also use $\boldsymbol{\varphi}$ to identify *splits* that link concavity pairs as discussed in the next section. When two concavities are linked in a split, the flow direction of the split is the flow direction of the concavities.

10.3 Splits

The previous section describes structures that capture important geometrical and topological aspects of a 2D glyph. These are the basis of a series of "divide and merge" operations that result in strokes. The divide operations, described in this section, indicate likely partitioning lines where the shape could be cut. The merge operations (Section 10.4) are based on identifying various types of "junctions" that reconnect parts to obtain plausible local stroke topology. These two complementary operations let us build strokes and use them for stylisations (Sections 10.5 and 10.6).

As we have seen in Chapter 3, pairs of concavities are important cues for the segmentation of object silhouettes into parts (De Winter and Wagemans, 2006) and many well-known approaches (Singh and Hoffman, 2001; Luo et al., 2015) use such pairs to define "partition lines" or "cuts" (Papanelopoulos et al., 2019) that delimit perceptually-distinct object parts. Because our goal is identifying strokes instead of general parts, we use related but different objects that we call *splits*.

A *split* delimits a protrusion in the outline with a line segment, which represents a potential location where strokes can cross or overlap. The line segment is constrained to be in the interior of the outline and links two concavities c_1, c_2 by connecting the extrema of their

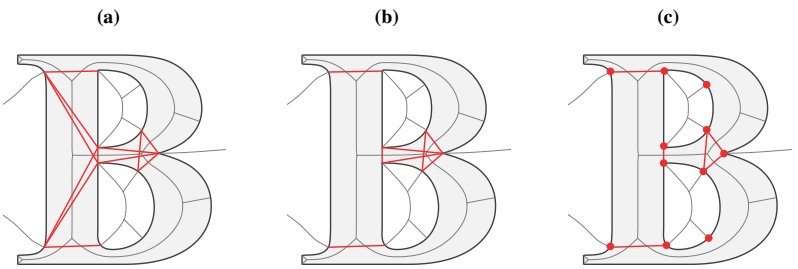


Figure 10.7: Valid and candidate splits selection. **(a)** Concavity pairs producing segments that are internal to the shape and intersect at least one SA^I_+ branch. **(b)** *Valid* splits, consistent with local conditions. **(c)** Final *candidate* splits based on saliency measure and organized in the graph GH. Note that GH also contains disconnected vertices for concavities.

CSFs. A split further identifies one SA^I_+ branch b_i that connects to a fork $f_j \in SA^I$. The fork f_j is always a fork in SA^I and never one introduced by the construction of SA^I_+ because those are associated with morphological features that do not indicate new strokes.

A split associates two concavities to determine a directional relationship between a fork f_j and one of its incident branches b_j. The branch protruding direction $\boldsymbol{\pi}(b, f)$ and the flow direction $\boldsymbol{\varphi}(c_i, c_j)$ are indicative of the direction in which a stroke should traverse the split. A single split indicates a *potential* separation or intersection of one stroke with another. Using good continuation to associate *pairs of splits* helps identify regions where strokes cross or overlap. Later, in the junction identification stage (Section 10.4), these pairs are used to link SA^I_+ branches across the region.

Valid splits Not all combinations of concavities produce a valid split and not all combinations of splits produce a valid segmentation (Figure 10.7.a, b). For two concavities to produce a valid split, they must be geometrically consistent with the definition above and they must obey a set of perceptually inspired local conditions (Section 10.3.1). As we detail in Section 10.3.2, the branch assigned to a split is not necessarily one intersecting it because of SA^I_+ distortions that often occur near ligatures. The validity of a split also depends on this assignment being possible, which is determined based on the configuration of SA^I_+ with respect to the outline.

Candidate splits and the split graph GH**.** Two valid splits might intersect, or might identify the same branch protruding from the same fork, in which case they are considered incompatible (Figure 10.7.b). We resolve these cases using a measure of *split salience* (Section 10.3.3) and only select the most salient split among incompatible ones, resulting in a subset of *candidate splits*. We then organize candidate splits into a *graph* $GH = (C, H)$ having one vertex for each concavity and one edge for each concavity pair connected by a candidate split (Figure 10.7.c). The connected components of GH define local areas where hierarchical relations

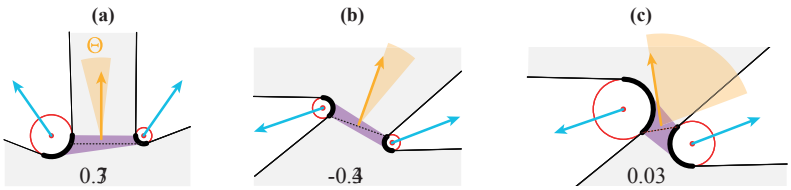


Figure 10.8: Computation of local convexity for different concavity configurations (red circles) and bisectors (blue arrows). The dotted black lines are the segments that maximize the left term of equation (10.4) over all segments (magenta area) that connect the contact arcs (thick black arcs) and the yellow arrows are perpendicular to them. The angle range $\bar{\Theta}$ covered by the segments is visualized as a cone in yellow. The examples **(b)** and **(c)** have the same contact circle centers and bisectors. With a "Z"-like configuration, the value of equation (10.4) in **(b)** is negative, thus invalidating the split. Given the tolerance ϵ_θ, increasing the curvature radii of the concavities in **(c)** results in a positive value and valid split.

between protrusions can exist (Macrini et al., 2011) or where split pairs can be associated. The graph is updated during the junction identification procedure (Section 10.4) and helps keep track of remaining splits and concavities to process.

10.3.1 Local conditions

We reduce the possible combinations of concavities inducing a valid split with two perceptually inspired constraints, one of *proximity* and one of *local convexity*. In our experiments these constraints improve the robustness of the method and avoid splits that would be otherwise perceived as delimiting invalid parts or protrusions of the outline.

10.3.1.1 Proximity

This is a basic Gestalt principle involved in perceptual organization (De Winter and Wagemans, 2006; Brooks, 2015). Since we assume the input is a combination of elongated strokes, we enforce proximity by examining the forks assigned to the split's concavities. For a split to be valid, either the concavities must be assigned to the same fork, or the distance between the fork centers must be less than a user-configurable multiple λ_r of the maximum disk radius in SA^I_+. We empirically find that a multiple of $\lambda_r = 3$ works well for our use case.

10.3.1.2 Local Convexity

Known to be an important cue in perceptual grouping (Elder, 2015), convexity has been used to drive a number of part decomposition approaches (De Winter and Wagemans, 2006). Similar to Papanelopoulos et al. (2019), we observe that for a split to produce a natural looking segmentation, it should delimit a region that is "locally convex" on at least one of its sides. *Local convexity* is achieved if both bisectors, one per concavity, are within some tolerance of pointing towards the same side of at least one segment connecting the two contact regions

associated to this pair of concavities (Figure 10.8). This holds if

$$\max_{\theta \epsilon \bar{\Theta}} \left[\sin(\phi_1 - \theta) \sin(\phi_2 - \theta) \right] + \sin^2 \epsilon_\theta \geq 0 \quad , \tag{10.4}$$

where $\bar{\Theta}, \phi_1$ and ϕ_2 are the angles that the segment and the bisectors make with respect to the horizontal, $\bar{\Theta}$ is the angle range spanned by the segments, and ϵ_θ is a user defined tolerance that we empirically set to 15°.

10.3.2 Fork and branch assignments to splits.

Splits induce parent-child relationships similar to the ones defined by Macrini et al. (2011), who describe a method to compute hierarchical relations between skeletal parts based on ligature analysis; in our experiments we found their method unable to handle the wide variety of situations typical of glyphs. Instead we use a heuristic method that is based on our outline analysis and produces more reliable results for our use case. The method depends on the enumeration of a number of branch and split configurations that we have found to occur in a variety of glyphs, and that are sufficient to produce plausible stroke segmentations for all the cases we have considered. A split is valid only if one of these configurations is identified.

We attempt to assign a branch b and fork f to each split, where f is one of the forks incident to b, f is near the split, and b, when considered to be starting at f, has a direction that indicates how the outline protrudes at the split. We formalize the notion of f being near the split by requiring it to be a member of F_C, the set of forks that have been assigned to either of the split's concavities (Section 10.2.1.3). We formalize the notion of the direction of protrusion by requiring that the dot product

$$d_\varphi(b, f) = \boldsymbol{\varphi}(c_1, c_2) \cdot \boldsymbol{\pi}(b, f) \tag{10.5}$$

be positive, where $\boldsymbol{\varphi}(c_1, c_2)$ is the flow for the split's concavities (Def. 10.2.3) and $\boldsymbol{\pi}(b, f)$ is the branch protruding direction relative to f (Def. 10.2.2).

There are five main cases to consider:

1. The split intersects one branch b (Figure 10.9.a). If b is incident to a fork f that is in F_C, and $d_\varphi(b, f)$ is positive, we assign (b, f) to the split.

2. The split intersects two branches b_1 and b_2 that are incident to the same fork f in F_C, and $d_\varphi(b_3, f)$ is positive for the third branch b_3 incident to f (Figure 10.9.b). If so, we assign (b_3, f) to the split.

3. The split intersects three branches that are incident to a fork f in F_C (Figure 10.9.c); this is the limit case of the previous configurations. We check the branch b incident to f that gives the largest value for $d_\varphi(b, f)$; if this value is positive, we assign (b, f) to the split.

4. The split intersects two branches b_1 and b_2 incident to the same fork f in F_C but $d_\varphi(b_3, f)$ is negative for the third branch b_3 incident to f (Figure 10.9.d). In that case, we consider f', the fork at the other end of b_3, and check whether f' is in F_C and $d_\varphi(b_3, f')$ is positive. If so, we assign (b_3, f') to the split. This represents a case similar to the first, but where SA^I_+ divides before the branch can cross the split. The division could indicate a stroke end, as shown in the figure, or it could continue into significant shape features, as in the middle of some letter K configurations. There is also a limit case for this configuration similar to (2).

5. The split intersects two or more branches that are incident to different forks in F_C (Figure 10.9.e). For each such fork f_i, we check the incident branch b_i with the largest value for $d_\varphi(b_i, f_i)$; if this value is positive, we compute the branch's salience $\beta_i = \beta(b_i, f_i)$ (Section 10.2.1.5). If none of these branches is salient, that is, $\beta_i < 0.5$ for all of them, this results in a special configuration we call a *compound split*. The split is assigned any one of these branches and its fork. This configuration is similar to (4), except that the protrusion is not sufficient for the branches to merge at a single fork.

In all other configurations, nothing is assigned to the split and it is rejected.

10.3.3 Split salience

The disambiguation of incompatible splits and the later junction analysis stage both rely on a measure of split salience. It uses four concepts from perceptually-driven studies of part decomposition to favor splits that are: (i) short (a.k.a. the "short-cut rule" (Singh et al., 1999; Singh and Hoffman, 2001)), (ii) connecting pairs of salient concavities (a.k.a. the "minima

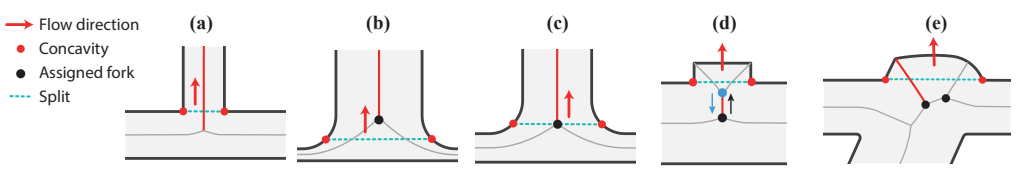


Figure 10.9: Branch and fork assignment of a split (dashed blue line), depending on its branch intersections. A red arrow shows the flow direction (Def. 10.2.3) of a split, while black dots and red segments are forks and branches that can be assigned to that split. **(a)** The split intersects one branch. **(b)** The split intersects two branches incident to the same fork, and the third branch leads into the outline protrusion. **(c)** Limit case between **(a)** and **(b)**, in which the split intersects all three branches incident to a fork. **(d)** The split intersects two branches incident to the same fork (upper blue circle), and the third branch (red) for that fork leads away (blue arrow) from the outline protrusion and is short enough that the fork at its opposite end (black circle) is in F_C; the split is assigned that (red) branch and (black) fork, since traversing the branch from the fork leads into the protrusion (black arrow). **(e)** The split intersects two non-salient branches incident to different forks, creating a compound split.

rule" (Hoffman and Richards, 1984)) (iii) located between outline regions with good continuation (a.k.a. "limbs" (Siddiqi and Kimia, 1995)) and (iv) separating a salient protrusion and branch (Hoffman and Singh, 1997). We have observed that each of these four concepts contributes to more robust segmentation results, which parallels human performances on part decomposition (De Winter and Wagemans, 2006). *Split salience* is thus computed as the sum of four terms:

$$\omega(\eta) = \bar{w} + \alpha + \beta + \lambda_\eta e^{(-\|\eta\|/r_{\max})} \tag{10.6}$$

combining the average salience value $\bar{w}$ of the split's concavities with the good-continuation value α between the concavities, the protruding branch salience β, and an exponential generalization function (Shepard, 1987) of the split length $\|\eta\|$ weighted by $r_{\max}$, the largest disk radius in SA^I_+. The latter term is weighted by a positive value λ_η that favors short cuts, which we empirically set to 2.

10.4 Junction Identification

Splits, together with CSFs, SA^I_+ and the associated information, provide a feature set that can be viewed as a geometric counterpart to representations that are hypothesised to occur preattentively in the human vision process. We use this feature set to construct a plausible graph structure for each stroke, which is achieved by organising all SA^I_+ forks and their connected branches into *junctions*. Formally, a junction J maps a set of forks F_J to a set of concavities C_J and splits H_J. Either C_J or H_J can be empty and for brevity we will say that the junction J *covers* the forks in F_J.

Junctions segment SA^I_+ by assigning *stroke labels* to its branches, with the labels being propagated across adjacent junctions and being determined by the junction type (Figure 10.10). There are six junction types organized in two main categories, topological (Ψ, Y and T), shown in Figure 10.11, and morphological (flexure, blunt-tip, null), shown in Figure 10.12.

Junctions are identified sequentially and each identification assigns labels to a group of branches incident to forks in F_J; these can either be new labels or labels already in the group. Figure 10.10.a shows a label from one junction propagating to a branch in a second junction. Branches can also be *multi-traced*; they accumulate labels and indicate a region where multiple strokes cross or overlap. In Figure 10.10.b the initial label of the central branch does not propagate to the vertical branches; instead the branch ends up with multiple labels.

Once all junctions have been identified and branches labelled, we use the labels to create *branch groups* (or groups for short): Subgraphs of SA^I_+ containing all branches that share a particular label. Two branch groups can share branches, for example, where one stroke crosses another. They form the basis for segmenting the glyph into strokes (Section 10.5). Note that SA^I is a subset of SA^I_+ so the procedure effectively segments both representations into strokes.

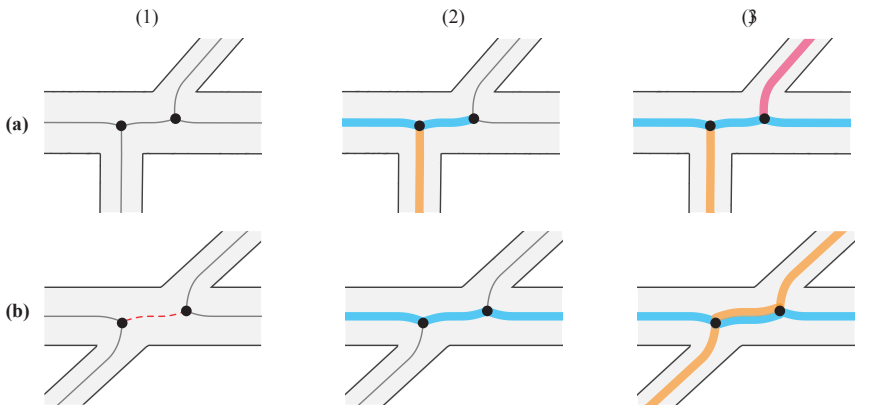


Figure 10.10: Label propagation in similar areas, each with two forks, but giving different branch groups. **Row (a)** (1): In this classification each fork will identify a separate T-junction, leading to three branch groups. (2): Classifying the first junction assigns one label to the vertical branch and a second label to the two horizontal branches. (3): Classifying the second junction also assigns a common label to the horizontal branches, propagating the previously present label to the second horizontal branch. **Row (b)** (1): In this classification there will be two Ψ-junctions sharing the same two forks, producing two branch groups that share a multi-traced branch (middle dashed line segment). (2): Classifying the first junction assigns the same label to all horizontal branches. (3): Classifying the second junction assigns another label to the vertically-oriented branches. Since the middle branch is multi-traced, its label is not propagated and it is assigned both labels.

10.4.1 Junction properties

Junctions uniquely determine the inferred stroke structure of an outline and are catgorised into two main types: topological and morphological. Topological junctions determine the connectivity between strokes, such as crossing, branching and incidence. Morphological junctions determine the local shape of a stroke, such as bends, corners or stroke-caps. This section describes the properties of the different junction types and how they label SA^I_+ branches, and the next describes how our system identifies these.

10.4.1.1 Topological junctions

Topological junctions come in three types: Ψ- Y-, and T-junctions.

Ψ-junctions: These occur when one stroke goes across one or more other strokes. A simple crossing like that in Figure 10.10.b consists of two Ψ-junctions that share the same forks; defining the junction this way simplifies the analysis of more complicated regions like the one in Figure 10.11.a. A Ψ-junction is characterized by a pair of splits that have opposite flow directions, are assigned to different forks, do not share a concavity, and have a high good-continuation value between the concavities at the split ends (Figure 10.11.a). Note that these splits cross the branch group and delimit where it enters and leaves the junction area;

they do not specify the cuts that will delimit the edges of the stroke as it goes through the area. The splits for the vertical blue path in Figure 10.11.a are the approximately horizontal splits indicated by the dashed blue lines. The *crossing path* is the shortest subset of SA^I_+ connecting the forks assigned to the splits. The domain F_J of the mapping that defines the junction, includes all forks along the crossing path. Each branch of the crossing path is designated as being multi-traced, and the crossing path and the two branches extending from the splits are assigned the same stroke label. A Ψ-junction does not label all branches incident to the forks in F_J; the remaining branches will be labelled later as being part of other Ψ-, Y- or T-junctions.

Y-junctions: Part of the shape branches out into two parts. A Y-junction is characterized by a salient concavity between two of the branches incident to a fork. We call this the *representative* concavity and call the branch opposite this concavity the *root* (Figure 10.11.b). There are four possible interpretations for a Y-junction: In the first two, one branch and the root share a stroke label, while one of the other branches is assigned another label. In the third one, the root is designated as multi-traced and assigned the stroke labels of the two strokes branching from it. In the fourth one, the three branches incident to the fork are assigned different labels. While this last configuration is valid, we chose never to use it based on a qualitative examination of the overall segmentation and stylisation results. The interpretation of a given Y-junction does not affect the identification of other junctions, but other junctions can affect the Y-junction interpretation, so we postpone the choice of interpretation until all other junctions have been identified.

T-junctions: Part of the shape protrudes in a near-perpendicular fashion. A T-junction is characterized by a *representative* split that separates the protruding branch from its originating fork (Figure 10.11.c). The junction assigns one label to the protruding branch and

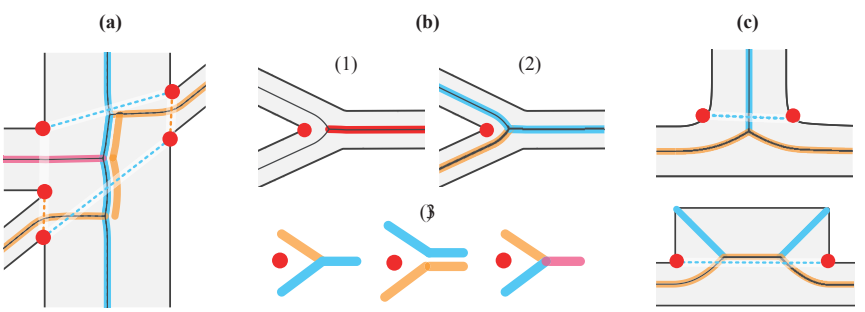


Figure 10.11: Topological junctions. **(a)** Ψ-junctions: Two Ψ-junctions, generating blue and orange branch groups, cover three forks. A T-junction, generating a magenta group, covers one fork. The crossing path is shared by both Ψ-junctions. **(b)** A Y-junction with (1) its salient concavity and root branch (in red), and (2) one possible branch group. Three other interpretations of the junction are possible (3). **(c)** T-junctions with (top) a single split and (bottom) a compound split.

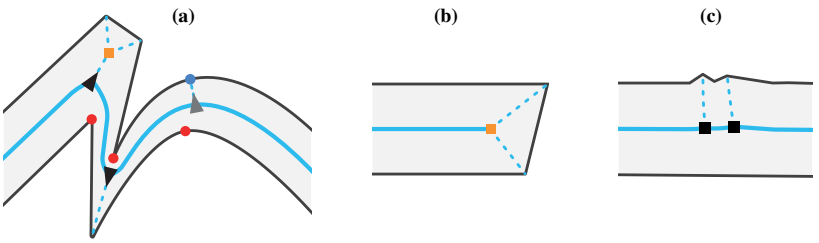


Figure 10.12: Morphological junctions indicated by their respective forks. Each example results in a single branch group. Dashed branches are disqualified. **(a)** Three flexures; two are strong (black triangles) and one is weak (grey triangle). All flexures are characterized by concavities (red circles) opposite the root branches. The weak flexure is characterized by a SA^I_+ branch that is not in SA^I. Note that the top flexure disqualifies a branch that terminates in a blunt-tip. **(b)** A blunt-tip marked with an orange square. **(c)** Two null junctions marked with black squares.

another label to all other branches incident to the forks covered by the junction – there can be more than one fork if the split is compound (Section 10.3.2).

10.4.1.2 Morphological Junctions

Morphological Junctions relate shape features of the outline, as indicated by CSFs, to forks of SA^I_+ while *disqualifying* non-salient branches. Disqualified branches are still labelled as being part of a group, but are ignored when we later turn groups into stroke paths (Section 10.5). Morphological junctions assign the same label to all branches incident to their forks and can occur in nested configurations (Figure 10.12.a).

Flexures: The shape contains a fork at a convex corner or bend that produces a branch in SA^I_+ (Figure 10.12.a). Flexures have a configuration similar to Y-junctions, having a significant concavity opposite a root branch, but with the root being *non-salient* with respect to the representative concavity. A *strong flexure* characterizes a corner or a sharp bend and is characterized by a root branch belonging to both $SA^I_+$and SA^I. A *weak flexure* determines a smoother bend and is characterized by the root belonging to SA^I_+ but not to SA. The junction disqualifies the root branch.

Blunt-tips: The shape contains a fork at a stroke end. A pair of non-salient branches protrudes from the fork and each such branch terminates at a convex CSF (Figure 10.12.b) or, in a tree-like branching hierarchy, near one. The junction disqualifies the two non-salient branches and any sub-trees that are present.

Null junctions: The shape contains a fork at a convex corner or bend that produces a non-salient branch in SA^I_+ with no opposite concavity (Figure 10.12.c). The junction disqualifies the least salient branch incident to the fork.

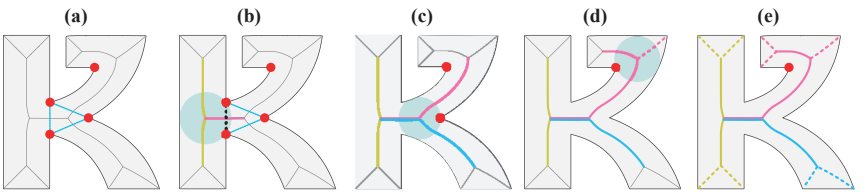


Figure 10.13: Iterative junction identification and stroke label propagation for a letter "K". From left to right: **(a)** The initial configuration of *GH* with vertices (concavities) in red, and edges (splits) in blue. **(b)** The first fork (blue disk) in the ordering results in the identification of a T-junction, induced by the black split. **(c)** One remaining concavity results in the identification of a Y-junction. **(d)** Another concavity produces a flexure. **(e)** The remaining forks produce blunt-tips.

ALGORITHM 1: Iterative junction identification

Data: SA^{I}_{+}, CSFs
Result: Junctions $\{J\}$
begin
  Assign concave features to forks (§10.2.1.3)
  GH = Split graph (§10.3)
  GX = Crossing graph (§10.4.3)
  for *split pairs* $(\eta_i, \eta_j) \in GX$, *sorted by decreasing* $\alpha(\eta_i, \eta_j)$ *(§10.4.3)* **do**
    if *no split in pair has been processed* **then**
      Label Ψ-junction
      Update splits and concavities in GH (§10.4.3.3)
    end
  end
  F = all forks not shared by two or more Ψ-junctions
  while F *is non empty* **do**
    Remove the fork with highest priority from F
    Label junction for fork
    Update splits and concavities in GH (§10.4.4.6)
  end
end

10.4.2 Iterative Junction Identification

Junctions often occur in complex and nested configurations and their identification becomes non-trivial. Similarly to existing approaches for part decomposition (Siddiqi and Kimia, 1995; Papanelopoulos et al., 2019), we resolve the identification problem with an iterative approach (Figure 10.13). Algorithm 1 gives an overview. We first identify all Ψ-junctions, since they define crossing paths through SA^I_+ that should not be disconnected by subsequently identified junctions. We then identify the remaining junction types.

10.4.2.1 Updating *GH*

We use the graph *GH* constructed in Section 10.3 to manage the changing configurations of candidate splits and concavities. Each iteration of the identification procedure removes vertices (concavities) and edges (splits) from *GH* depending on the identified junction. Removing a vertex also removes all incident edges, affecting the subsequent identification of remaining junctions.

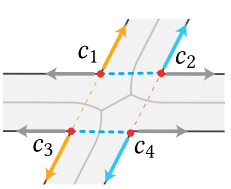
10.4.2.2 Labelling branches

Any existing label on a multi-traced branch is ignored, treating the branch as being unlabelled, since already-existing labels on multi-traced branches should not affect how other branches are labelled. Each junction identification uses the following rules to assign labels to groups of branches.

1. If no branch has a label, create a new one and assign it to each branch in the group.
2. If all labelled branches have the same label, assign that label to the other branches in the group.
3. If there are branches with different labels, arbitrarily choose one, assign it to all branches in the group, and also change all occurrences of unchosen labels in SA^I_+ to that label. This final change also applies to multi-traced branches.

10.4.3 Step 1: Identify Ψ-junctions

The identification of Ψ-junctions requires finding candidate split pairs in *GH* that can be associated based on good continuation.



For two splits η_i and η_j having concavities (c_1, c_2) and (c_3, c_4), we define the *connecting good-continuation value* $\alpha(\eta_i, \eta_j)$ to be the product, for the non-crossing split endpoint pairs (c_1, c_3) and (c_2, c_4), of the two good-continuation values (Note that these are different from the good-continuation values for the splits). When one of the splits is assigned a non-salient branch, we reorient the tangents corresponding to its concavities to match the split flow direction. This addresses one stroke crossing

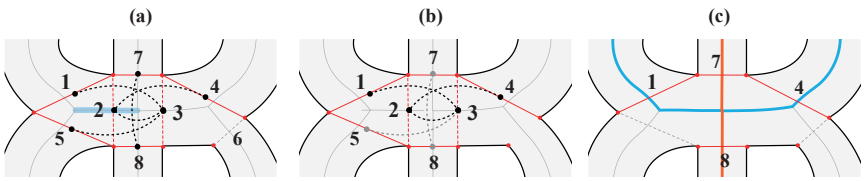


Figure 10.14: Ψ-junction disambiguation. **(a)** The red colored splits are all part of potential Ψ-junction pairs, which are edges in *GX* (dashed black arcs). The dashed-red splits are nested. For example split 2 is nested because its protruding branch (blue) is part of the crossing path between split 1 and split 3, which can be paired. Split 6 is not paired with any other split i because $\alpha(\eta_i,\eta_6)$ never exceeds the pairing threshold. **(b)** A path in *GX* connects splits 1,3,2,4. Splits 1 and 4 now form a new potential crossing pair. **(c)** This configuration results in two Ψ-junctions; note that there must be a short multi-traced branch where the branch groups cross. The remaining dashed grey splits will later identify T or Y junctions.

another but ending with a short, rounded protrusion (Figure 10.9.e), which sometimes occurs, for example in Chinese hanzi characters (Kishore, 2018). In this case the tangents do not adequately capture the perceived direction of stroke continuation.

Two splits can be paired if: (i) they are part of the same connected component in *GH*, (ii) they do not share a concavity, (iii) the connecting good-continuation value $\alpha(\eta_i,\eta_j)$ is greater than a threshold set experimentally to 0.15, and (iv) the segments connecting the split endpoints are well-aligned, which for our use case of fonts we interpret as the angle between them being less than 45°. Each such pair identifies a candidate Ψ-junction. Figure 10.14.a shows many potential pairings, but the only ones that meet all of these criteria are: $(1,3),(2,3),(2,4),(5,3),(7,8)$.

10.4.3.1 Nested splits and crossing graph *GX*

Candidate Ψ-junctions can occur in ambiguous nested configurations. We consider a split in a pair to be *nested* if it is assigned a branch that is part of any crossing path defined by another pair (Figure 10.14.a).

To identify valid Ψ-junctions and resolve nested configurations, we create an auxiliary *crossing graph, GX*, having one vertex for each split that is part of a pair and one edge connecting each pair. Nested splits will never be chosen to delimit a Ψ-junction but we still include them — they can act as a "bridge" that connects two other splits. In Figure 10.14.b splits 1 and 4 are not paired because their connecting good-continuation values are not high enough, but $(1,3),(3,2)$, and $(2,4)$ are valid, so 3 and 2 create a bridge connecting 1 and 4.

10.4.3.2 Identification

To identify crossing paths, we select split pairs among all pairwise combinations $\{\eta_i,\eta_j\}$ in *GX* where: (i) neither is a nested split, (ii) they do not share a concavity, and (iii) they are connected by a path in *GX* (Figure 10.14.b). At each step we select the one with the largest

cumulative product of the connecting good-continuation values $\alpha(\eta_m, \eta_n)$ for each split pair $\{\eta_m, \eta_n\}$ along the shortest path connecting η_i and η_j (Figure 10.14.c). In Figure 10.14.c we first choose $(7,8)$ because its crossing good-continuation value is very high. The remaining pairs $(1,4), (5,4), (1,5)$ are all connected by paths in GX but the pair $(1,5)$ is not valid because its splits share a concavity. We finally choose $(1,4)$, which has a slightly higher cumulative product than $(5,4)$.

10.4.3.3 Updating *GH*

Every time we identify a Ψ-junction, we remove its two splits from GH. We also remove any other split with an assigned branch that shares more than one vertex with the crossing path, which guarantees that the path is not disconnected by a subsequently-identified junction. For example, if the nested splits in Figure 10.14 were not removed, they would lead to T-junctions that would separate the crossing path produced by the split pair $(1,4)$. If a concavity is shared by two splits associated with different Ψ-junctions, we also remove it and any incident splits from GH.

10.4.3.4 Label assignment

After identifying a Ψ-junction we designate the branches on the crossing path as multi-traced and assign a single label to all branches.

10.4.4 Step 2: Identify Other Junctions

The five other junction types are assigned to one fork at a time. The identification of a junction for a given fork f depends on: the splits H_f that have been assigned to f, the concavities C_f assigned to f, and the salience of branches incident to f. Note that forks covered by a Ψ-junction can still have unlabelled branches (Figure 10.11.a).

10.4.4.1 Procedure

We process forks with unlabelled branches one at the time, in order of decreasing *priority*, given by:

$$\begin{cases} \max\limits_{\eta \in H_f} \omega(\eta)\,, & \text{if } H_f \text{ is non-empty}\,, \\ \max\limits_{c \in C_f} w(c) + \min\limits_{b \in B_f} \beta(b,f)\,, & \text{otherwise}\,, \end{cases} \tag{10.7}$$

where $\omega(\eta), w(c), \beta(b,f)$ are respectively the split, concavity and branch salience values of the splits H_f, concavities C_f, and incident branches B_f associated with the fork f.

This ordering favours processing forks with assigned splits before forks without a split, since the salience value of a split, $\omega(\eta)$, is the sum of four terms, including branch salience (equation (10.6)). In practice, prioritises T-junctions with high values of $\omega(\eta)$.

We distinguish junction types using the salience of the branches incident to the fork and the significance of the splits and concavities assigned to it. At each fork we construct a *branch salience ordering* to distinguish and disambiguate morphological junctions, and a

circular *significance histogram* with three sectors to disambiguate Y- and T-junctions, which are both described next.

10.4.4.2 Branch salience ordering

Morphological junctions involve the presence of one or more *non*-salient branches. Inspired by techniques in tensor shape analysis (Mordohai and Medioni, 2010; Westin et al., 2002), we detect different types of morphological junctions by sorting the saliencies of the branches incident to a fork (equation (10.2)) in decreasing order: $\beta_1 \geq \beta_2 \geq \beta_3$, and considering their relations normalized by the sum $\Sigma_\beta = \beta_1 + \beta_2 + \beta_3$. This gives three *junction classification measures*:

- $CM = 3\beta_3/\Sigma_\beta$: this distinguishes morphological from topological junctions. A value above a small tolerance τ_M, experimentally set to 0.2, means the least salient branch is still salient enough to indicate a T- or Y-junction or a flexure.
- $CB = (\beta_1 - \beta_2)/\Sigma_\beta$: a high value means that $\beta_1 > \beta_2 \simeq \beta_3$, suggesting a blunt-tip.
- $CF = 2(\beta_2 - \beta_3)/\Sigma_\beta$ a high value means that $\beta_1 \simeq \beta_2 > \beta_3$, suggesting a flexure or null junction.

Intuitively, this amounts to quantifying the shape of an axis aligned ellipsoid, with major axes of length $\beta_1, \beta_2, \beta_3$. Refer to Westin et al. (2002) for more details, including the reasoning behind the constant factors.

10.4.4.3 Significance histogram

We examine the influence on the fork of the concavities in C_f, as well as the ones associated with any split in H_f, by organizing them into three sectors. These are constructed by subdividing the plane with three rays going from the fork vertex to the three points where the fork's branches intersect the disk outline (Figure 10.15). If a branch does not intersect the disk, its ray goes through the tip of the branch. A concavity is assigned to a sector if its extremum is in the sector.

We compute the significance γ_i for each sector S_i, as:

$$\gamma_i = \sum_{c \in S_i} w(c) d(c, f) \quad , \tag{10.8}$$

that is, summing the significances of any concavity in the sector, which is defined as the concavity salience value $w(c)$ from equation (7.1) weighted by the concavity's influence $d(c, f)$ as defined in equation (10.1). If a sector contains no concavity, $\gamma_i = 0$. We sort the three sectors by decreasing order of significance values, $\gamma_1 \geq \gamma_2 \geq \gamma_3$, and consider their relations normalized by the sum $\Sigma_\gamma = \gamma_1 + \gamma_2 + \gamma_3$. This gives two *junction classification measures*:

- $CY = (\gamma_1 - \gamma_2)/\Sigma_\gamma$: a high value means that $\gamma_1 > \gamma_2 \simeq \gamma_3$, suggesting a Y-junction or flexure.

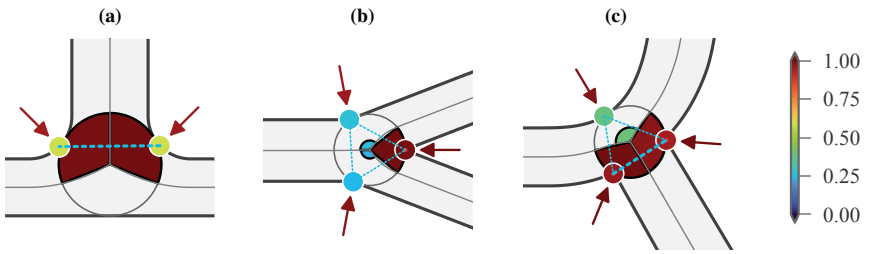


Figure 10.15: Significance histograms for different junctions. The bins, concavity saliencies, and concavity influences on the fork (visualized as arrows parallel to the bisectors) are colored based on γ_i, $w(c)$ and $d(c,f)$, with γ_i normalized to $[0,1]$. The bins are scaled proportionally to the respective value of γ_i. Splits are represented as dashed blue segments. **(a)** A T-junction with two concavities. **(b)** A Y-junction with three concavities and three splits. **(c)** A T-junction with three concavities and three splits.

- $CT = 2(\gamma_2 - \gamma_3)/\Sigma_\gamma$: a high value means that $\gamma_1 \simeq \gamma_2 > \gamma_3$, suggesting a T-junction.

For a Y-junction or flexure to exist, C_f must be non-empty, and the junction's representative concavity is the member of C_f with the highest value of $w(c)d(c,f)$. For a T-junction to exist, H_f must be nonempty, and the junction's representative split is the most salient member of H_f.

10.4.4.4 Identification criteria

The other five junction types are mutually exclusive and are classified using a *set of predicates* that depend on the branch saliency ordering, on the the significance histogram, and on the branches and concavities assigned to the corresponding fork. A predicate, denoted as $P(\bullet)$, identifies a property that distinguishes one or more junction types. The predicates are:

- $P(M)$: Indicates the presence of a morphological junction and is true if $CM \leq \tau_M$, which means there is at least one branch with low salience.
- $P(T)$: Suggests the presence of a T-junction because two sector significances are substantially higher than the third. $P(T)$ is true if one or more splits are assigned to the fork and $CT/(CY+\epsilon) \geq \gamma_T$, where ϵ is a small constant to prevent division by zero and γ_T is a user-defined threshold set to 0.2 in the examples given. Lower values of γ_T encourage identifying T-junctions, while higher values encourage identifying Y-junctions and flexures.
- $P(C)$: Is true if one or more concavities are assigned to the fork; that is, if C_f is nonempty.
- $P(B)$: Suggests the presence of a blunt-tip and is true if $CB/(CF+\epsilon)$ is greater than a user-defined threshold γ_B, experimentally set to 0.5 in the examples given.

- $P(R_\beta)$: Distinguishes Y-junctions from flexures and is true if there is a root branch b that is salient with respect to the representative concavity c, that is $\beta_c(b, f) > 0.5$ (Section 10.2.1.5)
- $P(R_I)$: Distinguishes flexures from blunt tips and is true if $CF > 0$ and the most salient concavity $c \in C_J$ has sufficient influence on the tip v_r of the root branch; in the given examples, we use $d(c, v_r) > 0.77$, corresponding to an angle of approximately 60°.

Connecting these predicates we can unambiguously decide the junction type with:

- *T-junction*: if $\neg P(M) \wedge P(T)$.
- *Y-junction*: if $\neg P(M) \wedge P(C) \wedge \neg P(T) \wedge P(R_\beta)$.
- *Flexure*: if $(P(M) \vee \neg P(R_\beta)) \wedge P(C) \wedge \neg P(T) \wedge P(R_I)$.
- *Blunt-tip:* if $P(B)$ and none of the above holds.
- *Null junction:* if none of the above holds.

10.4.4.5 Y-junction Interpretation

A Y-junction can be interpreted as either having: (i) a branching structure similar to a T-junction, (ii) a multi-traced root, or (iii) three separate groups (Fig. 10.11.b). However, we never choose the three-group interpretation based upon a qualitative examination of the stylization results. Then, the choice depends on the following multi-criteria condition: the configuration of the previously identified junctions, the local radius and protruding direction of each branch incident to the junction's fork, and on the length of the root branch. The local radius of a branch is given by the disk radius of the first vertex along the fork that is not part of a ligature. The ligature for the root is computed with respect to all concavities assigned to the fork, while the ligature for the other branches is computed only with respect to the junction's representative concavity.

The *degree of width disparity* between two branches is given by the ratio r_1/r_2, where r_1 and r_2 are the local radii of the branches sorted by decreasing radius. We define the *degree of alignment* α_b between two branches to be the good-continuation value between the protruding directions of the branches, with the vectors positioned at the vertices used to compute the local radii. The *relative root length* is defined as $(l_b - r_f - r_t)/r_f$ with l_b being the length of the root branch and r_f and r_t the disk radii of the root at the fork and at its opposite end.

In general, we find that the most effective stylizations are obtained by favoring the branching interpretation, reserving the multi-traced-root interpretation only for cases similar to the center of a letterform "B". However different threshold values can be used for different effects. The multi-traced interpretation is selected only if: (i) the root is the protruding

branch of a T-junction, or ends in a blunt-tip or a terminal vertex, (ii) the relative length of the root is less than a user-defined threshold Υ_l, and, (iii) the degree of width disparity between the two non-root branches is less than a threshold Υ_w. For our examples we use $\Upsilon_l = 3$ and $\Upsilon_w = 1.3$. Otherwise, we are in the branching configuration and we group the root with the branch for which $\alpha_b + 0.5(r_2/r_1)$ is highest.

10.4.4.6 Updating *GH*

As with Ψ-junctions, we remove vertices (concavities) and edges (splits) from *GH* after each junction identification. After identifying a T-junction, we examine each concavity connected by the junction's representative split. We calculate the absolute angle between the tangent on the concavity's side and the perpendicular to the flow direction of the split. If the angle is smaller than the local convexity tolerance ϵ_θ (Section 10.3.1.2), we remove the concavity and any split incident to it from *GH* (Figure 10.13.b). This is based on the observation that separating the protrusion identified by the representative split can produce a locally convex region in the neighborhood of the discarded concavity. We also discard a split from *GH* if it is incident to the representative concavity of a Y-junction or flexure.

10.4.4.7 Label assignment

After identifying a junction, we assign one or two stroke labels to all the branches incident to its fork. Morphological junctions assign a single label to all branches. T- and branching Y-junctions receive two labels, one to the two grouped branches and the other to the separate branch. Multi-traced Y-junctions also receive two labels, with the root receiving both.

10.5 From Junctions to Stroke Representations

Junction analysis merges SA^I_+ branches into a set of groups, one per stroke. For our use case of stylisation, we use branch groups to derive two stroke representations. First, a set of *stroke paths*, which are preferred paths extracted from each group, augmented with topological and morphological junctions and a varying width profile. Second, *stroke areas*, which are a set of potentially overlapping shape pieces that when unified closely reproduce the original glyph. Each representation enables different stroke stylisations , which are discussed in the following chapter.

10.5.1 Stroke Paths

A set of stroke paths enables many stylisation and animation effects because it captures a plausible way in which a glyph could be drawn. Each branch group is mapped to a unique path, which can easily be transformed into different kinds of strokes. Each path vertex maps to a SA^I_+ disk, which is used to assign the vertex a position and a width, and to annotate it with a junction if the corresponding disk is representative of one. The vertex positions and widths can be adjusted and are not necessarily the same as the corresponding SA^I_+ disk centers and radii, as we will see below.

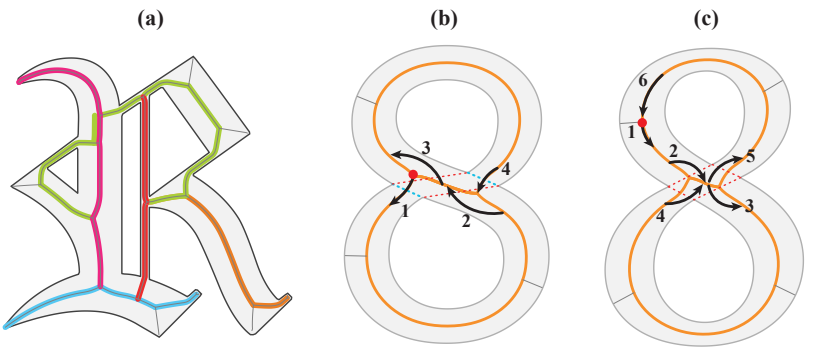


Figure 10.16: Stroke paths: Path construction from branch groups. **(a)** As long as no stroke graph connects to itself, we create preliminary paths by removing disqualified branches. **(b)** If a group connects to itself but is not closed, the path begins with a branch protruding from a T- or Y-junction. The dashed blue lines (1 and 4) are splits associated with a T-junction, while the dashed red lines (2 and 3) are splits associated with the Ψ-junction. **(c)** A different junction classification with two Ψ-junctions, leading to a closed branch group. In this case the path begins and ends at any group flexure, or at a null-junction if there are no flexures, or at an arbitrary vertex if neither is present. The arrows in **(b)** and **(c)** show how the path continues through vertices with degree > 2. Note that each constitutive SA^I_+ branch is part of at most one path unless it is multi-traced.

10.5.1.1 Path construction

We first transform the branching structure of each branch group into a preliminary path. We remove branches disqualified by morphological junctions (Figure 10.16.a), consider only one branch for compound splits, and traverse the rest of the graph using the connectivity determined by topological junctions (Figure 10.16.b, c). This results in a procedure that is effectively similar to the one that is typically used to "prune" symmetry axes (Shaked and Bruckstein, 1998), but it exploits junction analysis to determine which branches are non-significant. After this procedure, each path vertex still maps to an SA^I_+ disk, with its radius indicating the local thickness of a stroke, while a subset of these disks map to topological and morphological junctions. The path vertices and widths are adjusted based on this mapping.

Blunt tip and compound split adjustment. A blunt tip disqualifies two branches from a branch group and produces an end-vertex in the resulting path. The path usually ends too early with respect to the shape outline. We adjust the terminal vertex by moving it to the average position of the end points for the disqualified branch pair (Figure 10.17.a). For a compound split, we replace the multiple paths derived from the split's branches with a single straight one connecting the average of the split concavity extrema to the average of the split branch end points (Figure 10.17.b).

Ligature adjustment. The SA^I_+ disks for each resulting path are by definition maximal with respect to the outline, but their radii and the path itself can locally deviate from the perceived

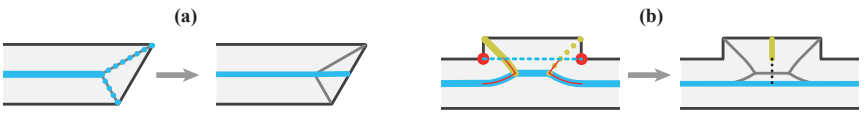


Figure 10.17: Adjustment of a blunt-tip **(a)**, and a T-junction with a compound split **(b)**. The blunt-tip disqualifies two of the branches incident to the fork (dashed blue) and the other branch is extended to reach the outline. The compound T-junction replaces the branches associated with the split with a straight path segment. The segment is extended (dashed black) to intersect with the opposite path.

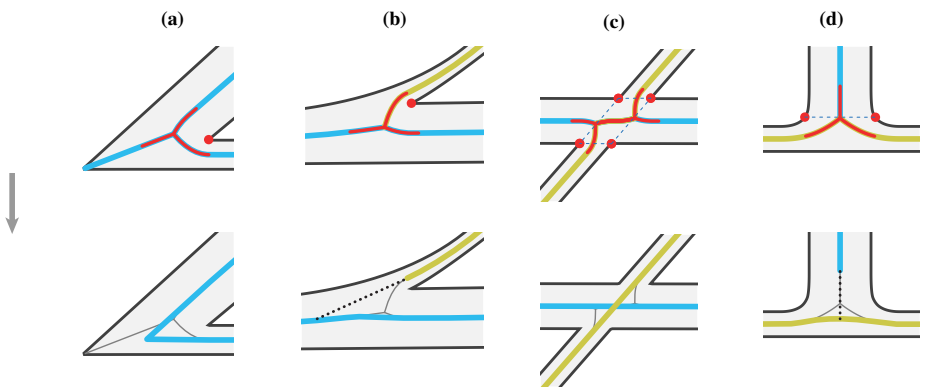


Figure 10.18: Examples of ligatures (top row: red segments) and their adjustments (bottom row: grey lines from discarded SA^I_+ segments, black dashed lines used as connectors), for the case of a: **(a)** flexure, **(b)** Y-junction, **(c)** Ψ-junction, and **(d)** T-junction. For the Y-junction **(b)** and T-junction **(d)** cases, the path is extended (dashed black) to intersect with the opposite path.

centerline and thickness of a stroke. This is especially apparent in ligatures near junctions, for example causing the zig-zags in the green path around the two loops in Figure 10.16.a. We follow a procedure similar to Macrini et al. (2011) and remove all path vertices that are part of ligature regions produced by the concavities associated with any junction falling along the path, unless the junction is a weak flexure. If this creates a gap in the path we close it in a way that depends on the junction type. If the junction is a flexure, we replace the removed vertices with a single vertex that maps to the flexure's fork and is located at the intersection of the tangent lines at the ends of the ligature segments. We set its disk radius to the maximum of the disks for these end points (Figure 10.18.a). Otherwise, we use the tangents to compute a cubic Hermite spline, sample it to create the vertices, and linearly interpolate the associated end point radii (Figure 10.18.b,d).

Path and width-profile smoothing. The procedures above can still result in paths and width profiles that contain undesirable variations. We remove such artefacts by smoothing the

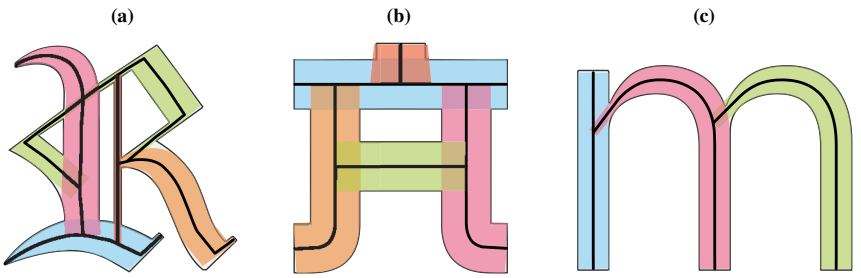


Figure 10.19: Adjusted stroke paths for three different glyphs. The spines, in black, are clean and plausible, and the union of stroke paths, in color, closely approximates its glyph outline.

paths and the corresponding radii in a piecewise manner, along path segments defined between adjacent end-points and strong flexures. This guarantees that path vertices corresponding to corners are not smoothed. In the examples given, we use a classic smoothing spline method (Dierckx, 1975). When a path segment is bounded by two endpoints or two elbows, we also check if it can be approximated with a straight line. To do so, we test the MSE of a linear least square fit to the segment vertices. If the MSE is less than a user-configurable ratio of the average path-segment width, we remove all the intermediate vertices resulting in a straight segment. This procedure is particularly useful to adjust the paths for short glyph parts such as *serifs*.

Path adjustment. Removing ligatures often disconnects T- and Y-junctions: we reconnect these by moving the path endpoint to the intersection of the path's end tangent and the other junction path (Figure 10.18.b, d, Figure 10.17.b). A similar process is used in Chapter 11 to reconnect paths for stylisation techniques that, while modifying paths, potentially disconnect these. Additional details are given in the next chapter. Figure 10.19 shows the final derived paths and width profiles for a few different glyphs.

10.5.2 Stroke Areas

Stroke areas are 2D shape pieces, where each piece is derived from a single branch group. Stroke areas enable stylisations that depend on the shape of the corresponding stroke. They are created by using *junctions* to partition the input shape into *disjoint faces* and then using the branch groups to guide the assembly of these faces into stroke areas.

10.5.2.1 Planar map $\bar{Q}$

We construct a *planar map*, $\bar{Q}$, from the glyph outline and from edges derived from the junctions. Each T-junction adds one edge to $\bar{Q}$, connecting the origins of tangents on the ends of its split (Figure 10.20.a). A multi-traced Y-junction adds two edges to $\bar{Q}$, each taking the direction of one of the tangents of the junction's representative concavity and connecting the concavity extremum to the first intersection with the outline (Figure 10.20.b). A branching

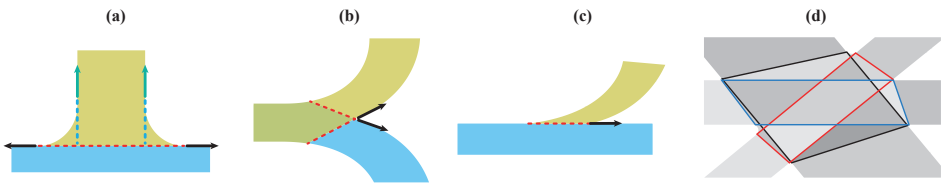


Figure 10.20: Faces and edges of $\bar{Q}$ for different junction types. The tangents determining the edges are marked in black. **(a)** A T-junction adds one edge and produces two faces, one including the two arc segments of the concavities' contact regions. **(b)** A multi-traced Y-junction adds two edges and produces three faces. **(c)** A branching Y-junction adds one edge and produces two faces. **(d)** Three Ψ-junctions in the same area, adding 12 edges (3 quadrilaterals).

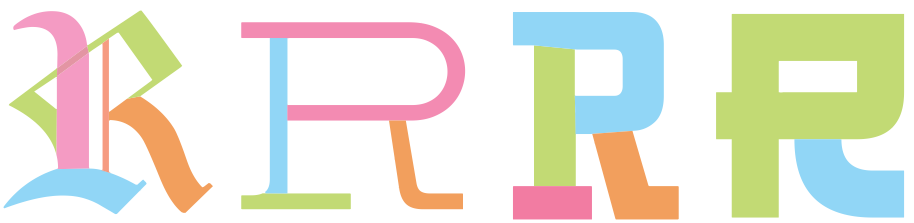


Figure 10.21: Stroke areas for the letter "R" found in four different fonts. Note that the last result (to the right) is based on a stroke group that contains a loop and crosses itself, producing a stroke area with a hole.

Y-junction adds one edge to $\bar{Q}$. If a split is assigned to the protruding branch, the junction is treated identically to a T-junction. Otherwise we take the direction of one of the tangents of the junction's representative concavity and connect the concavity extremum to the first intersection with the outline. The tangent is the one that is least aligned with the protruding branch (Figure 10.20.c). A Ψ-junction adds a quadrilateral to the graph. Two of its edges connect the tangent origins of the non-crossing split endpoint pairs, the same ones used to compute good continuation in Section 10.4.3. The other two edges connect the same tangent origins along the splits (Figure 10.20.d).

Once $\bar{Q}$ has been constructed, we create one stroke area for each group by performing a union of some of the faces in $\bar{Q}$. First, considering each group in turn, we construct an initial *seed area* by taking the union of all disks given by the vertices of that graph. We then assign to an area any associated face enclosed by the quadrilaterals added by a Ψ-junction to $\bar{Q}$. We also assign to a pair of areas any face associated with pairs of edges in $\bar{Q}$ linked to a multi-traced Y-junction. Finally, each remaining face is assigned to the area for which the intersection of the face and the seed area is largest. Fig. 10.21 shows the resulting stroke areas for the letter "R" in various fonts.

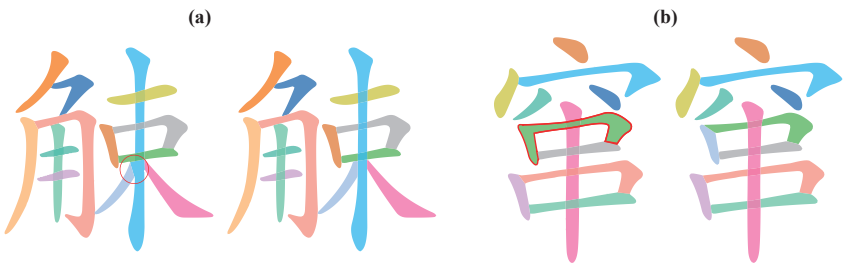


Figure 10.22: Quantitative evaluation with the *make-me-a-hanzi* dataset; ground truth is to the right. **(a)** Our method derives the same stroke structure as that of the ground truth but one T-junction (marked with a red circle) includes a stroke deformation. **(b)** All strokes are correctly identified by our method except for the middle area emphasized in red. We derive one single stroke rather than two as in the ground truth because there is no sufficiently salient concavity near the top left of that area.

10.6 Discussion and Results

Performance. The core segmentation procedure is written in the Python programming language, and it depends on the CSF code developed in Chapter 7. Outline analysis and segmentation together take an average of 2 seconds per glyph on an average laptop; normally we precompute these for an entire font, but they could also be computed on demand and cached.

Segmentation quality. Quantitative evaluation of the stroke segmentation results is difficult because of a lack of ground truth for Western fonts. However, we can compare the segmentation results with the *make-me-a-hanzi* dataset (Kishore, 2018), which includes outline and stroke ground truth for a variety of simplified and traditional Chinese characters. We tuned our parameters to give the highest accuracy for this dataset and then used these parameters for all our segmentations on other fonts and other objects. Similarly to Kim et al. (2018) we perform an "Intersection over Union" (IoU) test on the rasterised stroke areas. For each segmented stroke area, we identify the most similar stroke from ground truth by maximizing the intersection area. By rasterising at a resolution of 512×512 we achieve an average per pixel accuracy of 0.982, which is slightly better then the result of 0.958 reported by Kim et al. (2018), which is based on a neural network approach. This accuracy result is influenced by a few different stroke decompositions (Figure 10.22.b), as well as by a few inaccuracies in the estimation of planar map edges (Figure 10.22.a). We consider a stroke to be incorrect if its IoU is < 0.8, which does not include small errors like the one in Figure 10.22.a, and results in a per-stroke accuracy of 0.98.

It should be noted that we do not rely on training data and certain ground-truth decompositions cannot be deduced from the outline alone because they depend on domain knowledge. For example, "boxes" in Chinese characters should almost always be segmented

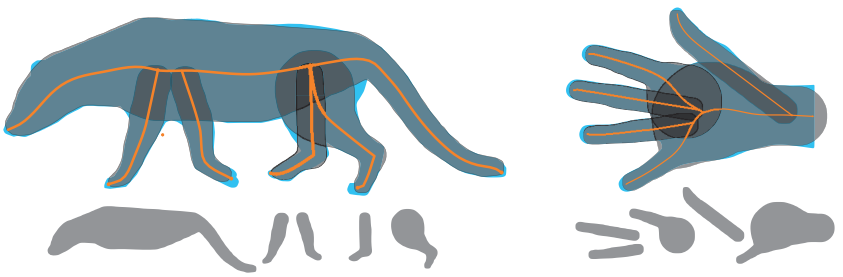


Figure 10.23: Stroke decomposition of silhouettes. The left mammal silhouette (from the PhyloPic database, `http://phylopic.org`) results in strokes that capture its articulated structure. The right hand results gives a plausible reconstruction, but the segmentation deviates somewhat from the perceived structure of a hand, e.g. with the pinkie being part of the same "stroke" as the palm.

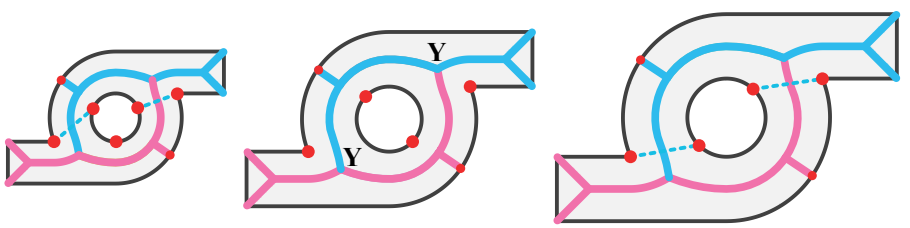


Figure 10.24: A glyph with a circular hole segmented at different scales. The circular region generates different CSFs depending on the scale, resulting in the detection of two T-junctions in **(a)** and **(c)** and two Y-junctions in **(b)**. However, because the Y-junctions are classified as branching, all scales result in the same, plausible stroke decomposition.

into three strokes. Sometimes there are outline details that lead to a correct segmentation, but not always (Figure 10.22.b). Only 5% of the glyphs in the *make-me-a-hanzi* dataset had segmentation errors that were not of this type; 10% had errors that could not be avoided without domain knowledge, and 85% were segmented identically to the ground truth. From a qualitative viewpoint, all of our segmentations (100%) produce strokes that create a readable reconstruction and robust stylisation of the glyph. Refer to Appendix D.2 for some additional examples from the *make-me-a-hanzi* dataset .

To explore further the generality of our approach, we have thus far tested our method on one hundred fonts. Appendix D.3 shows a number of example segmentations for fonts in different styles and languages. Plausible and useful segmentation results of individual glyphs are obtained in the vast majority of cases. The most common failure case is for very bold or thick glyphs in which the average stroke thickness is larger than the average stroke length.

Other considerations. The segmentation also gives useful results on other types of *non-glyph shapes* as long as there is a recoverable articulated or branching structure (Figure 10.23). We leave for the future work, the exploration, further development and comparison with other methods of our approach when applied to such objects.

As discussed in Chapter 7 the discrete Voronoi diagram is highly sensitive to circular or nearly-circular outlines, which can give results that vary with scale and sampling frequency (Figure 10.24). However, this potential issue does not seem thus far to have serious adverse effects on our segmentation results. Nevertheless, a possible alternative would be to simply combine the current Voronoi skeleton approximation of the SAT, when in the presence of nearly circular parts, with a robust method for the detection and representation of such shapes (Manzanera et al., 2016).

10.7 Conclusion

In this chapter we presented concepts and algorithms to automatically segment font glyphs into strokes. The segmentation relies on a geometric analysis of a glyph based on CSFs, and does not require training data. We used CSFs as the basis for implementing an experimentally validated model of good continuation and to derive two innovative representations, namely:

1. Splits that link concavities and describe potential shape divisions.

2. A set of six junction types that distinguish topological and morphological structures.

The extraction of a stroke structure from a glyph outline can be used to generate a variety of stylisations of the input (Figure 10.1), which can be explored in real time by a user or designer. The next chapter will demonstrate a few of these use cases, and in particular stylisations that exploit the stroke representations developed in this thesis. This segmentation could also be useful in related applications like automatic font hinting (Shamir, 2003), segmenting characters in historical documents (Lamiroy et al., 2015), painterly applications of robotics (Deussen et al., 2012), stylisation methods that require taking glyph structure into account (Zou et al., 2016), and animated reconstructions of arbitrary glyphs (Gingold et al., 2008).

The current trend in the computational science field, is to solve these kinds of segmentation and stylisation problems with a data-driven approach, often with a preference for end-to-end solutions combining one or more statistical models. These methods typically rely on a large body of human-labelled training data. We instead demonstrated a solution that relies on experimentally-validated principles of visual perception and computational geometry concepts. The advantage of our approach is that it is adaptable to fonts for which training data might be scarce or non-existent and to glyphs that do not match the training data. Our solution requires tuning a few parameters, but these have intuitive visual and perceptual interpretations and can be adjusted by the user for the required use case.

In future research, we plan to explore how data-driven solutions could be combined with our approach. For example, we could use data to incorporate language-specific domain knowledge. More fundamentally, we could use a data-driven approach to set parameters in the junction identification stage. We hypothesize that training on a very small number of user-labelled examples could be enough to create a mapping between forks, their associated CSFs, and the six different junction types we introduced. Developing a similar procedure with automatically labelled examples is also an interesting avenue of future research. While ground truth data for Western fonts is scarse, a stroke-based font description language such as METAFONT (Knuth, 1999) can be used to automatically generate parametric variations of strokes, as well as variations of the glyph outlines resulting from their combination. The resulting training pairs could be used to drive a fully automatic data-driven solution to the stroke segmentation problem. Finally, the method presented in this chapter already has potential uses in producing reliable training data for sequence-based generative models like the one developed by Ha and Eck (2018) for SVG drawings. While similar methods have been successfully used to reproduce handwriting (Graves, 2013), or Chinese characters (Tang et al., 2019), to the best of our knowledge this approach remains to be further developed and tested with a variety of fonts and styles.

Chapter 11

Font stylisation

Figure 11.1: Stylisation of the word GRAFFITI generated from recovered stroke paths using our methods for segmentation (Chapter 10) and strokes (Chapter 6).

Stroke paths and areas (Chapter 10) are the basis for a variety of stylisation methods. Grounding text stylisation on fonts has the advantage that it is agnostic to the language or writing system and the embedded kerning information can be used to determine inter-glyph spacing, which is known to be difficult to achieve with methods that create stylized text from scratch (Haines et al., 2016).

While the previously described segmentation procedure runs offline, the stylisation procedures described in this chapter run in real-time and support the exploration of different stylisations through an interactive user interface. All the procedures are written in C++ using OpenGL for hardware-accelerated rendering.

11.1 Path-based stylisation

Stroke paths capture a plausible way in which a glyph could be drawn and provide a rich structural description that enables a range of stroke-based stylisations. These range from

Figure 11.2: Hershey font stylisation (black) overlayed on the original font (gray).

design-oriented stylisations based on skeletal strokes, to stylisations that resemble calligraphy and graffiti art, which exploit the stroking methods described in the previous chapters.

11.1.1 From stroke paths to strokes

The first step to construct a stroke stylisation from a set of stroke pathsis to augment these paths with a linear vertex-ordering. While for certain stylisations an arbitrary ordering can be sufficient, many other types of strokes are not necessarily invariant to the direction of traversal. The following examples all use a simple topological sorting heuristic that rewards top-to-bottom and left-to-right movements. However, the path representation is also suitable for more sophisticated approaches (Fu et al., 2011; Tang et al., 2017). The topological sorting procedure results in a sequence of densely sampled polylines. Each polyline vertex maps to

Figure 11.3: Font stylisation with skeletal strokes. The left column shows the text in the original font. The right column shows the corresponding stroke stylisations. The first example on the right shows the result of using skeletal strokes as implemented in Adobe Illustrator to change weight, cap, and join styles; the other three show various decorative effects. The strokes in the last example use variable width.

a path vertex, together with the corresponding width and junction annotation if present.

The initially dense polyline representation can be used to produce some simple stylisations. For example, we can convert it into spines consisting of Bézier curves, which can then be used to generate "Hershey fonts", which have glyphs consisting of constant-width strokes (Figure 11.2). Such fonts are well-suited for fabrication and manufacturing applications. The same spines can be used to construct skeletal strokes (Section 3.6.2) (Asente, 2010), which enable a variety of glyph stylisations ranging from the more painterly to the more decorative effects (Figure 11.3).

11.1.2 Simplification: constructing motor plans

The stroke stylisation methods described in the previous chapters are designed to work with motor plans consisting of relatively sparse polylines. We convert an the initially dense representation into a sparse one by using polyline simplification. The simplification can be done with a variety of methods (Luebke, 2001), but we choose to use Discrete Contour Evolution (Latecki and Lakämper, 1998) (Figure 11.4.a), which selectively removes polyline vertices based on a circular arc-length *relevance measure* (Table 3.5, Chapter 3). A vertex is removed if its relevance is less than a user selected threshold and we never remove vertices corresponding to strong flexures by assigning these a maximum relevance. Because the simplification method removes vertices, the mapping from the remaining spine vertices to junctions and widths is always maintained.

Adjustment. After simplification, we adjust the spine endpoints to the closest intersection of end-tangents with the opposite spine, with a procedure identical to the one used when constructing stroke paths. The stylisation procedures that follow also require a similar adjustment, with the addition of a few steps that depend on the method and are discussed

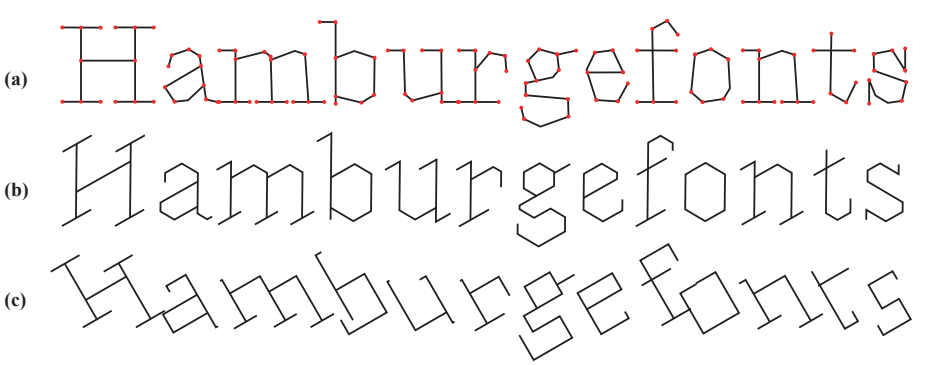


Figure 11.4: Simplification and schematisation: **(a)** Path simplification (kept vertices as red dots). Spine schematisation (Dwyer et al., 2008): **(b)** quantising orientations to multiples of 60°, and **(c)** restricting orientations to 30° and 120°.

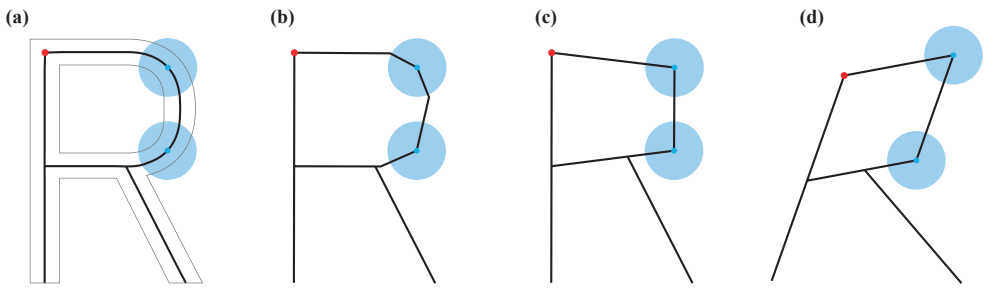


Figure 11.5: Mapping to flexures (blue circles) for **(b,c)** simplified and **(d)** schematised spines.

when relevant.

11.1.3 Structural modifiers

The connectivity information encoded by topological junctions can be used to transform a motor plan through *structural modifiers* that enable glyph stylisations that resemble graffiti or calligraphy, while taking the glyph structure into account.

11.1.3.1 Schematisation

Schematisation quantises the orientations of spine segments and results in regular-looking polygonisations and stylistic abstractions of a glyph structure (Figure 11.4.b,c). These kinds of regular structures can be observed in some graffiti letter stylisations. Ferri (2016) includes a similar construct in his "form functions", which he hypothesises underlie the genesis of graffiti styles (refer to Table B.2, F7). We implement schematisation with a so-called $\mathscr{C}$-oriented method (Nöllenburg, 2014), which approximates a polyline with another one consisting of segments that are parallel to a discrete set of orientations $\mathscr{C}$. We use a least-squares solution to the $\mathscr{C}$-oriented problem proposed by Dwyer et al. (2008) for a metro-map generalisation task.

The schematisation procedure can alter the number of vertices in a spine and this can corrupt the mapping from spine vertices to flexures, which is necessary to drive the subsequent stylisation procedures. To recover this mapping we compute a set of correspondences between the schematised and non-schematised spine vertices (Figure 11.5). For this purpose, we use Dynamic Time Warping (Gold and Sharir, 2018) with the Euclidean distance between vertices, and then assign a flexure to a schematised vertex if it is assigned to any corresponding non-schematised vertex.

Structural adjustment. Schematisation is applied to each spine separately which can corrupt the motor plan connectivity, making it difficult to apply intersection-based adjustments to the spine endpoints. While a correct topology could be imposed with constraint solving algorithms (Nöllenburg, 2014), we observe that this issue mostly affects spines such as the

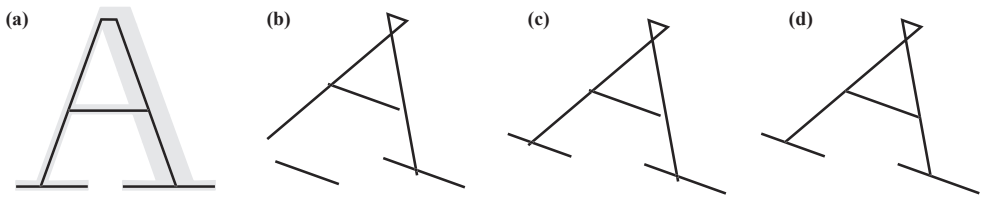


Figure 11.6: Structural adjustment steps for a schematised letter "A". **(a)** Unstylised stroke spines after reconstruction. Note that incidence relations for T-junctions have been already corrected as described in Figure 10.19. **(b)** Schematisation can corrupt topological relations among strokes. **(c)** We re-establish these by shifting strokes that are covered by a single T-junction or branching Y-junction. Note that the triangular part of the "A" is covered by two T-junctions, so it is not adjusted. **(d)** A second adjustment step reconnects all stroke endpoints.

lower-left serif in Figure 11.6.b, which is characterized by another spine ending within it. This kind of configuration can be detected by counting the number of T-junctions and branching Y-junctions along a spine. If, for a given spine, only one such junction exists, we translate the spine by $\boldsymbol{p}' - \boldsymbol{p}$, where $\boldsymbol{p}$ is the original endpoint of the incident spine and $\boldsymbol{p}'$ is the endpoint after schematisation (Figure 11.6.c). Once this procedure is executed, we can adjust the endpoints with the same tangent intersection procedure as before (Figure 11.6.d).

11.1.3.2 Structural decorations.

Topological junctions are also useful to identify spine segments that can be altered for additional stylisation effects. In one such method, we extend spine end segments that do not terminate in any topological junction (Figure 11.7.a), by a configurable amount that is proportional to the length of the segment projection on the horizontal or vertical axis. Figure

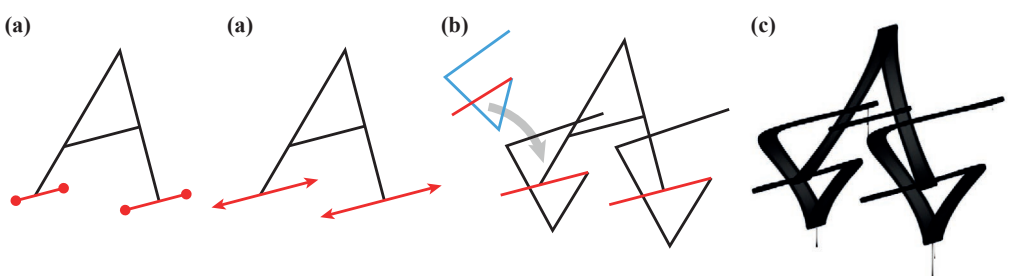


Figure 11.7: Structural decoration steps for a schematised letter "A". **(a)** Schematised "A". The two serifs end-points do not terminate in any topological junction. **(b)** Extending the serifs for effect. **(c)** Decorating the serifs with a user defined motor plan (top left), that replaces the serif red spine segment (in this case the whole spine). The replacement is made by rotating the motor plan so that its first segment (also in red) matches the replaced spine segment. **(d)** Example calligraphic stylisation of the resulting motor plan.

11.7.b shows how this method can be used to extend the serifs of a glyph. In a second method, we replace these end segments with a user-defined motor plan (Figure 11.7.c), resulting in a procedure that is similar to a shape grammar (Stiny and Gips, 1972). The motor plan is transformed so that its first spine segment matches the end segment that is replaced. This method can be used to mimic the "flourishes" that sometimes adorn calligraphic letterforms or similar decorative elements that can be observed in graffiti stylised letters (Figure 11.7.d).

11.1.4 Calligraphic Stylisation

The simplified and structurally modified motor plans can be used to construct a variety of calligraphic stylisations that mimic the aesthetics of certain kinds of calligraphic writing (Figures 10.1.e,f and 11.8) or graffiti tags (Figure 11.11).

We generate kinematic realisations of the motor plans using MIC with semi-tied covariances and one Gaussian component for each motor plan vertex. Each component is scaled by a factor in $[0,1]$ that depends on whether a vertex maps to a flexure or not, which allows to produce a variation of curvature that is similar to the original glyph. If a vertex maps to flexure the scaling factor is given by:

$$\frac{\max(r, r_{\max})}{r_{\max}} \quad ,$$

where r is the radius of curvature associated with the flexure and $r_{\max}$ is a user-configurable

Figure 11.8: Calligraphic stylisations generated by combining schematisation (Dwyer et al., 2008) with kinematic realisations generated with MIC (Chapter 5) and kinematics-based brush rendering (Chapter 4).

Figure 11.9: Calligraphic stylisations of the string "AUTOGRAFF" with brush thickness proportional to the path width profiles and using schematisation with multiples of 45° starting from a user selected initial orientation (different for each row).

Figure 11.10: Painterly stylisation of the Chinese string "qiyun shengdong" (left), using multiple overlapping skeletal strokes constructed along kinematic realisations of the schematised spines.

maximum radius value. If the vertex does not map to any flexure, the factor is set to its maximum value of 1. Similarly to Chapter 5, different kinematic realisations of a motor plan are produced by also globally varying the scale, isotropy and the orientation of the covariance ellipses.

Different combinations of structural modifiers, kinematic realisations and stroke rendering methods (discussed in Chapters 4 and 5) can be varied interactively, resulting in stroke stylisations that resemble instances of calligraphy (Figures 11.9 and 11.10) or graffiti tags (Figure 11.11).

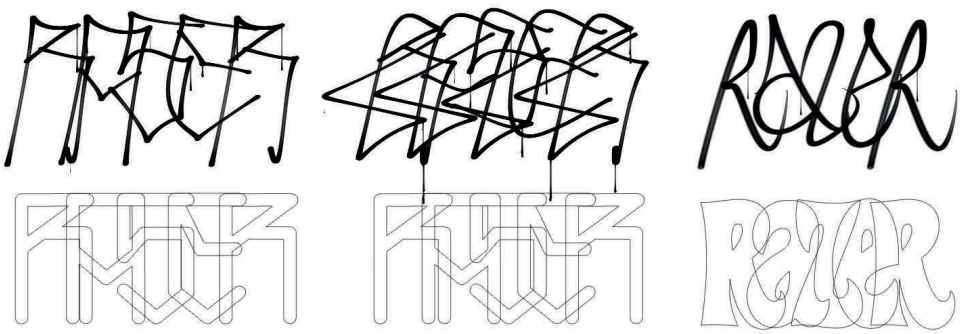


Figure 11.11: Three different tag-like stylisations of the word “RASER”; below each is the corresponding font and glyph-spacing. Note that, the middle stylisation is done by replacing near vertical spine segments with a user-defined motor plan.

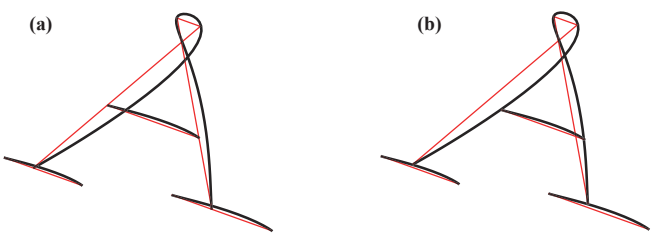


Figure 11.12: Structural adjustment steps for a kinematic realisation of a schematised “A”. **(a)** Using the adjusted schematised motor plan (red) from Figure 11.6 to produce a kinematic realisation (black) can also corrupt the incidence relations among strokes. **(b)** A last adjustment step moves the non-smoothed spine endpoints (the middle section of the "A") so they terminate at the intersection with the smoothed trajectory.

11.1.4.1 Smoothed stroke adjustment.

Smoothing also can corrupt the adjacency relations between strokes (Figure 11.12.a). To adjust these configurations, we perform a first smoothing pass on each spine resulting in an initial set of trajectories. We then adjust the end-vertices of the spines so that their endpoints are incident to these trajectories (Figure 11.12.b). Finally we perform a second smoothing pass on the adjusted spines.

11.1.5 Outline Stylisation

A similar procedure to the one above can be used with the outline based stroking method described in Chapter 6, resulting in glyph stylisations that mimic the appearance of graffiti pieces (Figures 11.17 and 11.13). The local and global smoothness is determined similarly to the calligraphic stylisation case, but we also generate a piecewise-smoothed stroke if the

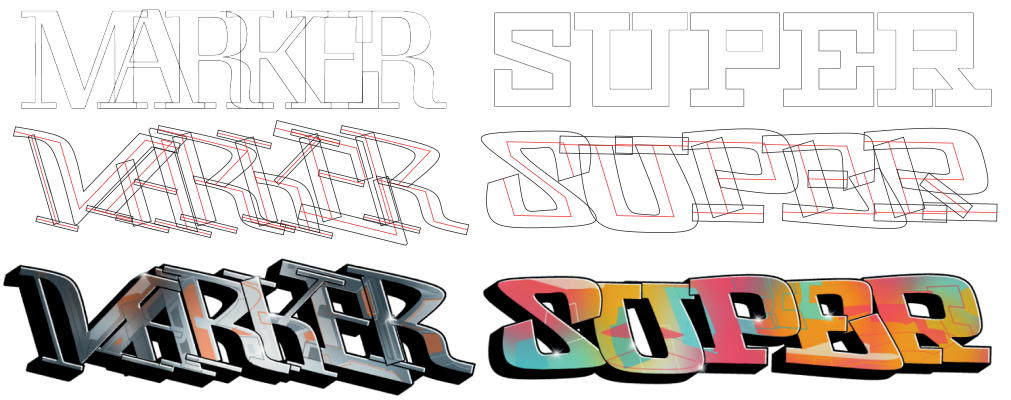


Figure 11.13: Outline-based graffiti stylisation. *Top:* the source fonts. *Middle:* strokes and schematised spines (red). *Bottom:* Layered and rendered graffiti stylisations.

smoothness of a vertex is below a user defined threshold (Figure 11.14). This allows us to reproduce strokes that combine smooth and straight portions and to keep corners with increased levels of smoothing.

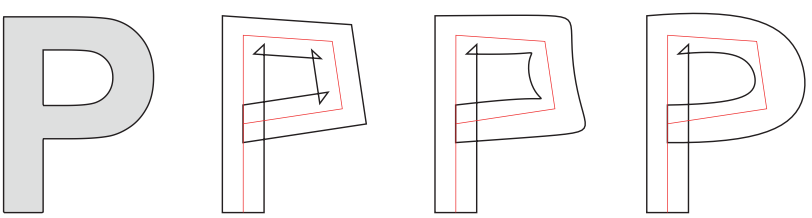


Figure 11.14: Progressive smoothing of a letter "P" (Arial) with a corner and consisting of a single stroke. The corner is maintained across increasing levels of smoothing.

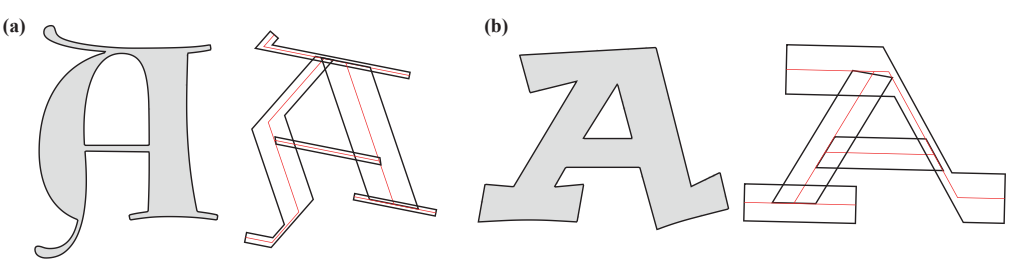


Figure 11.15: Per-segment width profiles. **(a)** The width of each simplified segment is proportional to the average width of the intermediate *scaffold*path vertices. **(b)** Parametric width profile depending on the direction of the spine segments.

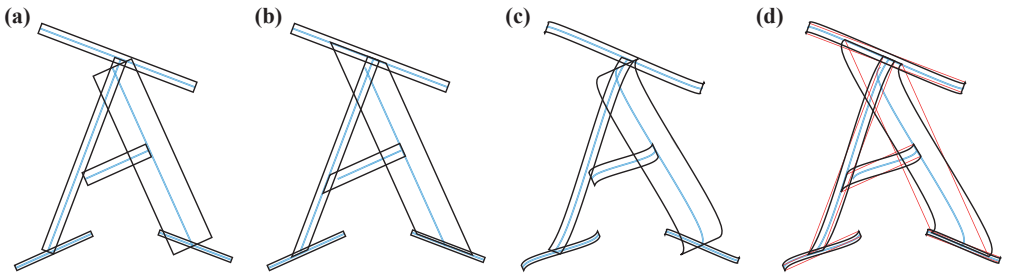


Figure 11.16: Structural adjustment steps for the outlined strokes of a schematised "A". **(a)** Without adjustment, the sides of the strokes can terminate outside of the stylised glyph area. **(b)** The stroke sides are adjusted so they terminate at the intersection with a thinner version of the opposite stroke. **(c)** The same issue for smoothed stroke outlines. **(d)** For the smoothed case, the polygonal outline (i.e. the motor plan, in red) is adjusted, resulting in new trajectories that terminate at the intersection with the thinned (and smooth) opposite stroke.

The width profile for the strokes can be computed from the scaffold or parametrically. When using the scaffold we assign each spine segment a constant width, given by the average width of the scaffold vertices spanned by the

The width profile for the strokes can be computed from the corresponding paths or parametrically. When using the paths, we assign each spine segment a constant width, given by the average width of the path vertices spanned by the segment (Figure 11.15.a). Otherwise, the width profile can be computed parametrically with one of the methods described in Chapter 6, e.g. based on the orientation of each spine segment (Figure 11.15.b).

The local depth ordering of strokes is either determined randomly or with a point-and-click procedure identical to the one described in Chapter 6. We automatically add local unions for partitions that coincide with topological junctions, but these can also be disabled for different stylisation effects.

11.1.5.1 Outlined stroke adjustment.

The adjustment procedure is slightly different for outline based strokes (Section 6.1.1), since the thickened stroke sides can overshoot the opposite strokes (Figure 11.16.a). To adjust these configurations we first duplicate all the strokes and scale the corresponding width profile by an arbitrarily small amount. We then adjust the end-segments of the polygonal sides so their end-vertices are incident to the opposite scaled stroke (Figure 11.16.b). Similarly to the case of calligraphic stylisation, when a stroke outline is a smooth trajectory (Figure 11.16.c), we adjust the corresponding polygonal outline, which results in a different and adjusted trajectory (Figure 11.16.d).

Figure 11.17: Combining schematisation with the outline-based graffiti strokes (Chapter 6) to generate graffiti stylisations of strings in different fonts and languages.

11.1.6 Stroke animation

The topologically sorted strokes can be easily animated with a variety of methods. Calligraphic strokes can easily be rendered and animated with the same techniques discussed in Chapters 4 and 5. This results in natural-looking animations that reflect kinematics that are similar to a human hand motion (Figure 11.18). Stylized brush animations can also be gen-

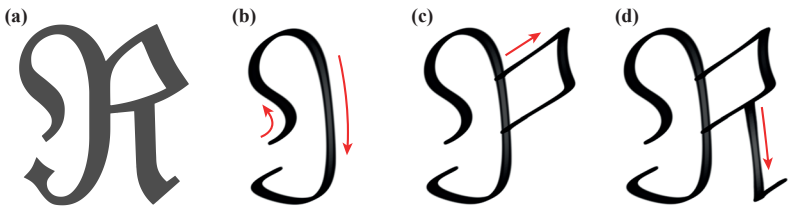


Figure 11.18: Animating the drawing of a stylized "R".

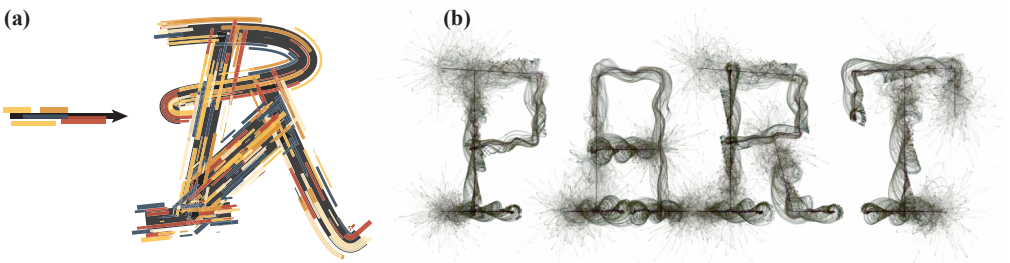


Figure 11.19: Abstract stroke-based animations. **(a)** Applying an animated prototype (left) to a styled letter "R". **(b)** Particle animation following the stroke spines and leaving traces.

erated by incrementally visualizing a skeletal stroke, animating the skeletal stroke prototype itself (Figure 11.19.a), or by animating a particle system that follows the stroke spines (Figure 11.19.b).

Stroke areas can also potentially be used to create an animated reconstruction of the glyph with an automatic version of the template-based method developed by Gingold et al. (2008). We did some preliminary tests in this direction, but the development of a working prototype is left as future work.

11.2 Area-Based Stylisation: Stroke Similarity

The stroke area segmentation of the outline is the basis of a similarity measure among strokes in a complete font. We compute the difference between two stroke areas by aligning their centroids, rasterising them, and then measuring the *Jaccard distance* (Deza and Deza, 2013, p. 299) between the resulting bitmaps: i.e. 1 minus the intersection divided by the union. If one stroke terminates in a topological junction and the other does not, the distance takes the maximum value of 1. We then group strokes using single-linkage agglomerative clustering (Murphy, 2012) and determine clusters based on a user-configurable threshold. While the distance is computed offline, the clustering procedure is interactive, and users can adjust the threshold to their preference. We then replace each stroke area in a cluster with an artistic rendering based on the shape, generating stylisations that apply uniformly across an entire font (Figures 10.1.i and 11.20).

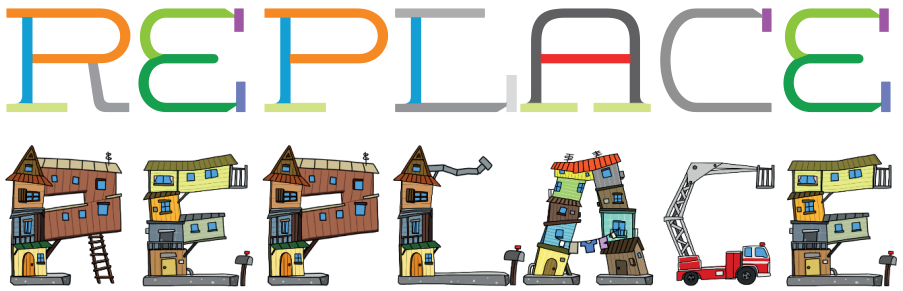


Figure 11.20: Stylisation based on similarity between stroke areas. In the first row, strokes are color-coded based on common clusters. In the second row, each stroke in a cluster is replaced with the same custom artwork. Note that including junction structure in the stroke similarity metric allows distinct stylisations to apply to otherwise similar strokes, like the horizontal strokes in R, P, L, and A. Artwork ©Daichi Ito.

11.3 Conclusions

In this chapter, I have demonstrated how the stroke segmentation procedure developed in Chapter 10 can be used to generate a rich variety of stylisations and animations of a font. The range of stylisations includes design-oriented stylisations based on skeletal strokes, schematic abstractions of a glyph structure, calligraphic or graffiti stylisations and similarity-based replacements of parts of a glyph.

The proposed methods achieve one of the objectives set in the introduction (Chapter 1) of generating graffiti stylised strings in a variety of languages and styles. By exploiting the latent stroke structure encoded by font outlines, the proposed solution transforms a much more challenging problem of glyph synthesis (Hofstadter et al., 1993) into a simpler one of inverse modeling and stylisation, where the wealth of publicly available fonts becomes a rich source of possible glyph structures.

The resulting system allows a user to rapidly generate and customise high quality graffiti textures, which can be easily applied to the surfaces of a computer generated environment. Figure 11.21.a shows examples of the output of this system textured in an environment that can be explored in real-time with the Unreal® game engine. Currently, this requires the user to manually apply the textures where desired, but developing automatic methods to do so is an interesting avenue of future research. Likewise, the tag textures shown in the environment are currently static, but animating their reproduction with a full-body inverse kinematics procedure is possible and another promising research avenue (Figure 11.21.b). Another interesting application to explore is the use of augmented reality (AR) technologies to apply the generated graffiti to real-world walls (Figure 11.21.c).

[1] `https://github.com/Squashwell/bepuik/tree/bepuik`
[2] `https://apps.apple.com/us/app/wallr`

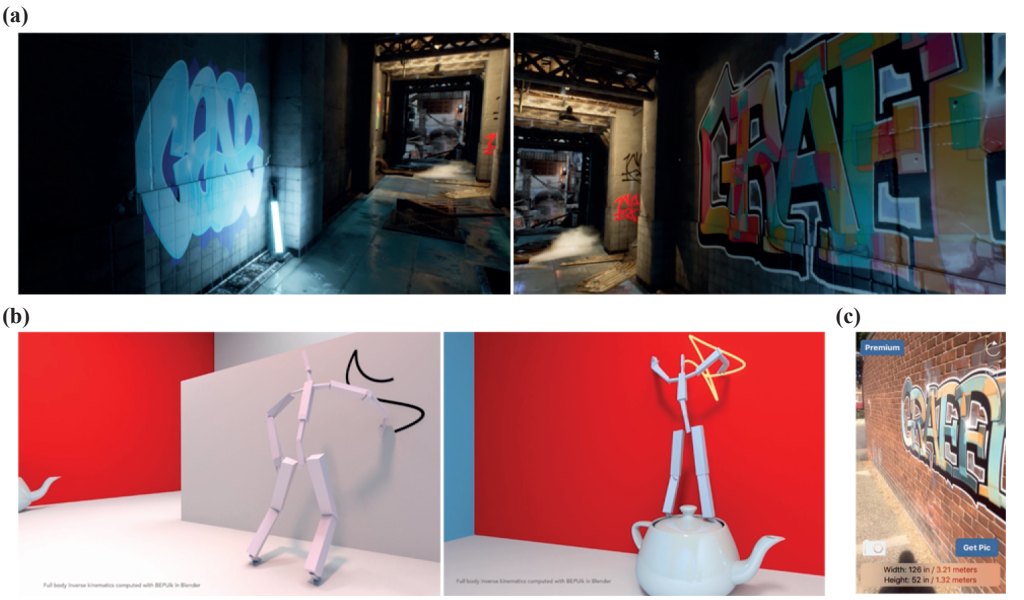


Figure 11.21: Synthetic graffiti in the virtual and real world. **(a)** Textures generated by our system applied in a virtual environment within the Unreal game engine (by Epic Games). **(b)** Prototype for full-body inverse kinematics animation of the production of a tag. Given a hand trajectory, the full body inverse kinematics are computed using the Bepuik[1] tool for the Blender 3D package. **(c)** AR graffiti experiment using the WALLR[2] application.

While the system is capable of generating convincing graffiti stylisation, this still requires an effort on the part of the user to appropriately choose a set of stylisation parameters that work well with a given font or combination of glyphs. An evaluation of the quality of the stylisation techniques is beyond the scope of this work. Our main goal was to enable a wide variety of stroke-based stylisations and the proposed implementation provides a "sandbox" in which the user can explore many different options in real time. These range from readable stylisations to highly abstract renditions that are still evocative of the original font structure, but that are difficult or impossible to read. This applies especially to the calligraphic and graffiti stylisation methods, which operate in a domain where aesthetics take priority over readability (Craveiro, 2017).

Chapter 12

Conclusion

In this thesis, I have presented a series of primitives, methods and tools meant to computationally reproduce the appearance of graffiti art as well as certain related forms of calligraphy. The work initially stems from my previous experience as a graffiti artist. It builds on a personal introspection into a design process that I have assimilated over the years, but also on feedback and knowledge gained from peer graffiti artists, some of whom have shared their ideas in the form of published books (Ferri, 2016; Arte, 2015; Kimvall, 2014). While being informed from my personal experience and intuitions, the implementation of the methods developed across the thesis is grounded on an in-depth study of methods, principles and results in a number of fields, including computer graphics, computational motor control, graphonomics, visual perception and shape analysis.
In the introduction, I did set two principal thesis objectives:

1. Implementing a system that enables a user to rapidly "sketch" graffiti in a variety of styles and with a user interface that is similar to the one typically used in standard vector-drawing applications.

2. Generating high quality graffiti content that can be customised by a user and can be textured in virtual environments in games and in movies.

These objectives are intended to address the current lack of existing research or methods that are aimed at the computational generation of graffiti art.

The resulting solutions are also conceived to address a number of related challenges and limitations that I have identified in the attempt to design graffiti in conventional vector-based design applications, or while observing graffiti art as it can be seen textured in video games and movies; namely:

1. The difficulty to specify, edit and manipulate calligraphic curves, such as the ones that are typically seen in graffiti art, with conventional curve generation and editing methods.

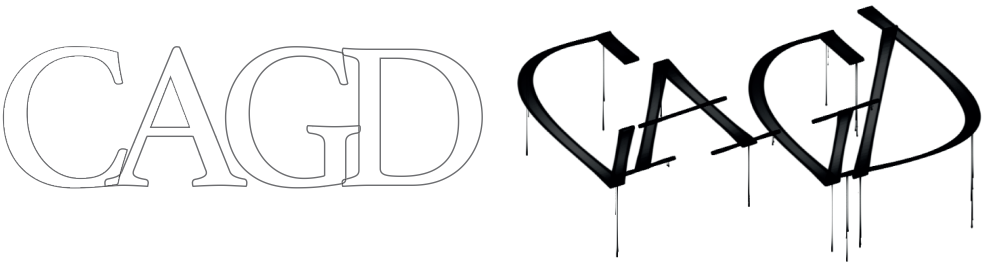


Figure 12.1: Calligraphic stylisation of the word "CAGD".

2. The difficulty to reproduce the often self-overlapping and intertwined patterns that can be observed in graffiti pieces, with the standard back-to-front object layering typically assumed in vector-design packages or methods.

3. The low realism, variation and customisability of graffiti that can be seen textured in computer generated environments.

The methods discussed in the thesis are developed for the specific use case of graffiti art. However, I have shown that reaching the objectives above, results in methods that are generally useful in a broader design spectrum, with applications in calligraphy, typography and pattern design. The resulting system is meant to be compatible with standard CAGD pipelines, leading to the same acronym but a more specific goal of "Computer Aided Graffiti Design" (Figure 12.1).[1] According to the prefixed objectives, I have organised the thesis into two main parts, the results of which results are briefly summarised and discussed next.

12.1 Part I: Stroke primitives

To describe a variety of graffiti styles with a similar framework, I have proposed a a two-level representation of stylised letterforms, consisting of a motor plan and a set of stroke primitives.

The motor plan is a schematic representation of a series of idealised movements that trace a stylised version of the letterform. Strokes materialise these movements with stylised traces or outlines, finally resulting in different kinds of letter stylisations. The motor plan is similar to certain end-point representations of movement (Plamondon, 1993; Maarse, 1987) that are hypothesised to occur in the brain (Flash and Hochner, 2005). It consists of a sparse sequence of points, that is easy to specify and edit, similarly to the control polygon typically used to specify curves in standard CAGD applications. It can be constructed manually, with a simple point-and-click procedure, or automatically from a set of traces (Chapter 8) or from the outlines of a glyph (Chapter 10 and Chapter 11).

[1] While still keeping geometry in the loop.

I have emphasised early on that the fundamental "atom" of graffiti art is the tag, a highly stylised signature, the visual quality of which is directly related to the spontaneity and skill with which its drawing movements are executed. In order to reproduce the "hand style" that is used to produce tags I proposed a "movement centric" approach to curve generation and a related concept of "style by kinematics", in which different stylisations of a curve are produced by varying the parameters of a movement that follows a common motor plan. I have demonstrated two possible implementations of this approach: one, the $\Sigma\Lambda$ model (Chapter 4), based on a space-time superposition of ballistic movement primitives, and a second, MIC (Chapter 5), based on a probabilistic formulation of optimal control. Both methods produce movement trajectories with kinematic properties that are characteristic of human hand or arm movements, such as a high degree of trajectory smoothness (Engelbrecht, 2001; Sosnik et al., 2004), bell-shaped speed profiles (Morasso, 1981; Plamondon et al., 1993) or an inverse relation between trajectory speed and curvature, *i.e.* isogony (Viviani and Terzuolo, 1982; Lacquaniti et al., 1983). These properties are useful to generate natural stroke animations, or to vary brush thickness or density in order to mimic ink deposition during drawing movements. Furthermore, the parameterisation of both models implicitly defines variations of a trajectory that reproduce the variability that is typically observed in multiple instances of human writing or drawing. I have demonstrated examples of how these properties can be advantageous to reproduce instances of graffiti tags and how similar results are difficult to achieve with conventional curve generation methods such as Bézier curves.

In Chapter 6, I demonstrated how the same movement centric approach is useful to generate outline-based strokes, which can be combined to reproduce the appearance of a diversity of graffiti styles other than tags. Again, the method mimics a sketching procedure that might be followed by a graffiti artist while tracing a stylised stroke outline. Different types of strokes and stylisations are achieved by varying the way in which these tracing movements evolve. This outline based stroke representation also enables self-overlaps as well as local layering and union effects that are difficult to achieve in conventional vector-design packages (Asente et al., 2007; McCann and Pollard, 2009), and this facilitates the reproduction of these same visual qualities that can be often observed in graffiti art.

I have justified the choice of a movement centric approach to curve generation with a hypothesis of "embodied aesthetics"(Freedberg and Gallese, 2007) suggesting that the observation of a static trace, such as the mark left by a brush, produces a recovery of a likely generative movement in the viewer. Based on commonly held knowledge in the graffiti art community (Ferri, 2016; Craveiro, 2017), but also among calligraphers (Briem et al., 1983; Wang, 2013), I have adopted the hypothesis that the qualities of a latent generative movement also influences the aesthetic appreciation of the resulting trace. Based on this hypothesis, I have suggested that the same should hold for the traces of a computer generated movement. While this remains a conjecture, the methods developed in this thesis have enabled an on-

going series of experiments (Chamberlain et al., 2019, 2020) , which suggest the validity of this hypothesis, at least for the case of "expert" viewers with a prior experience in the arts. An overview of these early results is given in Section 12.4.4.

12.2 Part II: Graffiti content generation

In the second part of the thesis I have exploited the primitives developed in the first, in order to develop a system in which a user is able to (i) vary and stylise existing traces of tags or handwriting and to (ii) generate and customise graffiti text strings in a variety of styles and with arbitrary languages and writing systems.

To achieve the first goal, I developed a method that recovers a motor plan and a set of kinematic ($\Sigma\Lambda$) parameters from from the geometry of a static trace (Chapter 8). This reconstruction effectively separates a structural (motor plan) and kinematic (remaining $\Sigma\Lambda$ parameters) component from a given trace geometry, which enables a user to edit, vary, render and animate the trace with procedures that are similar to the ones discussed in Chapter 4. I also demonstrated how this separation can be exploited to implement a novel example-driven stylisation procedure (Chapter 9) that is conceptually similar to a number of "style transfer" methods that rely on some form of separation between a descriptor of "style" and a descriptor of "content" (Hertzmann et al., 2001, 2002; Li et al., 2013; Gatys et al., 2015). The novelty of the proposed approach is the use of kinematics as a descriptor of (hand) style.

The second goal of generating graffiti strings in different styles and languages was perhaps the most challenging problem in this thesis. In his book *Metamagical Themas* (1985), Douglas Hofstadter has pointed out the difficulty of this kind of problem, which he compared to "knobbifying" the alphabet, that is finding a general and finite set of parameters that can be used to reproduce the enormous variety of possible structures and styles that a letterform can take. Rather than solving this ill-posed problem, which pertains more to the fields of computational creativity (Boden, 2003) and cognitive science, I have proposed a still challenging but more pragmatic solution: recovering useful strokes from the outlines of a given font (Chapter 10). While the proposed solution might leave some unsatisfied, it effectively tackles the letterform generation problem by exploiting the wealth of publicly available fonts as a source for possible letter structures and styles.

Both stylisation methods described in the second part of the thesis rely on the recovery of stroke primitives that reconstruct a given input geometry, which functions as a "seed" that generates diverse instances of synthetic graffiti. Seeking ways to solve this reconstruction problem has led to the development of curvilinear shape features (CSFs), a novel "contour + boundary" shape descriptor inspired to the work of Leyton (1988) and based on the computation of local symmetry axes (Chapter 7). CSFs lead to the definition of the CASA, an extended version of Blum's *SA* (Blum, 1973) with branches terminating at all curvature extrema. In Chapter 8, I've shown how CSFs can be used to accurately identify curvature ex-

trema and to infer a plausible generative movement for a given trace, by exploiting the $\Sigma\Lambda$ parameterisation together with the isogony principle. In Chapter 10, I've shown how CSFs, together with the CASA, can be used to characterise glyph outlines with a perceptually inspired measure of good continuation and to segment these outlines into potentially crossing or overlapping strokes. I expect to identify many more use-cases for CSFs in my future research, and I hope that this representation will prove useful to others, beyond the scope of this thesis.

12.3 Summary of Contributions

The main contribution of this thesis is a set of tools and a methodology for the computational reproduction of graffiti art. In general, this adds a previously neglected art form to the ones that have been studied in the computer graphics literature and in the growing field of NPAR (Kyprianidis et al., 2013). From a practical standpoint, the resulting system is designed to fit into a standard vector-design pipeline and to let a user focus on high-level compositional aspects, while the system takes care of producing outputs that are similar to graffiti art.

At the same time, the objective to model this art form has led to a number of contributions, the utility of which extends to the wider domains of computer graphics and even more generally to computational shape analysis:

- A movement centric approach to curve generation aimed at reproducing hand-drawn or written traces, taking into account well known properties of human hand/arm movements (Chapters 4 & 5) and consequently enabling realistic modeling of variability, natural looking stroke animations and kinematics-based rendering of strokes.
- A semi-tied covariance formulation for the generation of calligraphic stylisations with optimal control and a reduced number of open parameters (Chapter 5).
- An extension to skeletal strokes that enables self-overlaps and smooth outlines that mimic the ones produced with a drawing movement (Chapter 6).
- Curvilinear Shape Features (CSFs), a novel representation that describe convex and concave outline or contour features (Chapter 7).
- Curvilinear Augmented Symmetry Axis (CASA), an augmented symmetry axis that include features missed by the traditional definition according to Blum (Chapter 7).
- A novel example-driven curve stylisation method that uses movement kinematics as a feature representation and as a descriptor of style (Chapter 9).
- A novel geometry-based segmentation method that can decomposes fonts into potentially crossing and overlapping strokes (Chapter 10).

More generally, the proposed subject of study has proven to be useful beyond the initially set artistic and design-oriented goals. Graffiti art is a peculiar art form because it revolves around stylisations and customisations applied to letterforms, with a varied but well defined set of visual conventions and procedures (Kimvall, 2014; Ferri, 2016), involving the execution of skilled movements, and where letterforms are perceptual units that involve both shape (Sanocki, 1992) and motor (James and Gauthier, 2006) representations in the brain. As previously mentioned, the methods and ideas developed in this thesis have contributed to the development of a series of ongoing experiments in the field of empirical aesthetics, aimed at studying and understanding the links between the kinematic qualities of drawing movements and the perceived aesthetic quality of the resulting traces. The results of the experiments have contributed to support the hypotheses underlying this thesis, and the methods developed in this thesis have contributed in systematically generating stimuli for the experimental procedures. I argue that this kind of cross-disciplinary feedback loop can contribute to the development of more accurate and sophisticated computational models of a given art form while contributing, at the same time, to a better understanding of the complex mental and physical processes involved in art-making.

12.4 Limitations and future work

The presented work is not free from limitations, and the most important ones are summarised and discussed next. Those limitations also open up a number of paths for future research, which are also discussed in the following sections.

12.4.1 $\Sigma\Lambda$ model

Concerning the $\Sigma\Lambda$ model, I made the choice of keeping the lognormal shape parameters μ_i and σ_i fixed to predefined values, given the observation that they produce a negligible effect on the trajectory geometry. However, the inconclusive results discussed in the conclusion of Chapter 8 suggest that a more rigorous analysis of these parameters' effects on trajectory kinematics is useful , especially in sight of comparisons with other physiologically plausible movement models such as minimum jerk (Section 8.4). One way to determine the parameters is to first (i) reconstruct various digitised tags with one of the existing kinematics-based $\Sigma\Lambda$ parameter estimation methods (O'Reilly and Plamondon, 2008; Plamondon et al., 2014; Fischer et al., 2014; Ferrer et al., 2018) and then (ii) compute μ_i, σ_i from the average of the parameter estimates. However, it should be noted that these methods also make specific assumptions when reconstructing a trajectory, and these assumptions are likely to produce differing values of μ_i and σ_i.

As discussed in Chapter 4, the $\Sigma\Lambda$ trajectory generation method still lacks a principled way to explore different stylisations of a motor plan with few parameters, such as the approach we have seen with MIC together with semi-tied covariances (Section 5.2.2) or multiple

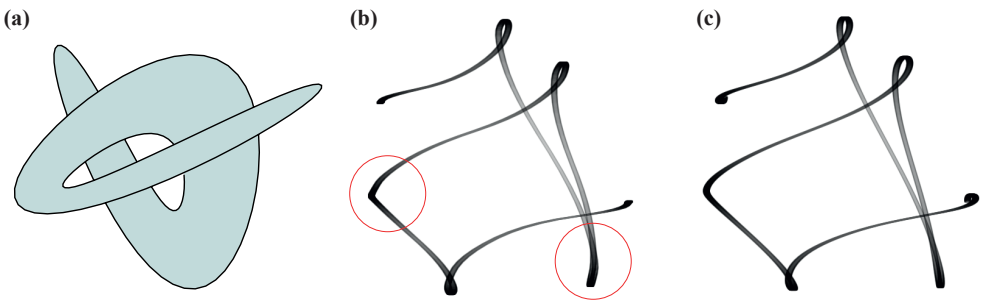


Figure 12.2: Comparison of dynamic B-splines with MIC. **(a)** An outline based stroke (Chapter 6) generated with the dynamic B-spline method of Shinoda et al. (2003). Comparison of **(b)** a calligraphic trajectory generated with the method of Shinoda et al. (2003), and **(c)** a similar trajectory generated with MIC and a 3rd order system. The two trajectories use the same semi-tied covariance structure. Note that the B-spline appears to have a lower degree of smoothness near the loci circled in red.

references (Section 5.1.7). The possibility to efficiently compute key-points along a trajectory (Section 4.4.2.1) together with high-order derivatives at their time occurrence (Section 4.1), suggests that an optimisation-based approach is a potential solution to this limitation. The combination of optimal control with the $\Sigma\Lambda$ model is generally a promising research avenue. For example, in Section 8.4, we have seen that the overlap between lognormals produces a delay between velocity minima that is similar to the the one predicted by the minimum jerk model (Flash and Hogan, 1985; Todorov and Jordan, 1998). This suggests that optimising the time-overlaps between lognormals can result in trajectories that are optimal in a minimum-square-derivatives sense (Engelbrecht, 2001), and this can potentially lead to an efficient solution to the uniform parameterisation problem that affects MIC similarly to splines (see Section 5.4.2 and Lee, 1989).

12.4.2 MIC

The main limitation of the MIC method (Chapter 5) is the inefficiency of the batch solution, which requires solving a potentially large linear problem with $O(n^3)$ computational complexity. While the iterative solution described in Appendix C.2 is much more efficient (Figure 5.27), it does not enable some useful functionalities, such as stochastic sampling and periodic trajectories. The possibility to refine an initially sparse trajectory estimate mitigates the issue with the batch solution, but the computational complexity of the method still limits the use of MIC as an interactive curve generation tool in vector-design applications.

One way to improve performance is to formulate the optimisation problem in terms of B-splines, as proposed by Fujioka and colleagues (Shinoda et al., 2003; Fujioka et al., 2006; Fujioka and Miyata, 2011). The B-spline formulation still requires the solution of a $O(n^3)$

problem, but it does not require considering full states (with all derivatives) for each time step and thus results in a smaller optimisation problem and a more efficient solution. As a preliminary test, I implemented the method of Shinoda et al. (2003), extending it with a weighted formulation in terms of semi-tied Gaussians (Section 5.2.2). While the method provides a valid replacement for MIC in the outline based case (Figure 12.2.a), the results appear less satisfactory for the use case of calligraphic stylisation (Figure 12.2.b and Figure 12.2.c). However these results are purely qualitative and deserve a more in depth comparison of the advantages or disadvantages of both methods. In general, certain contributions such as the semi-tied covariance formulation presented in Chapter 5 are independent of the optimisation methods used, and exploring different basis function representations of the problem (including lognormals) is an interesting avenue of future research. The control-based approach used for MIC is not ideal in terms of efficiency, but it is highly flexible. For example, the same optimal control problem can be also formulated in the frequency domain (Calinon, 2019), which has interesting implications when modeling oscillatory behaviors in drawing or scribbling movements.

12.4.3 Graffiti design

One aspect that is not taken into account in this thesis is the emergent character of the graffiti drawing/writing process. With the proposed methods, graffiti letter design is reduced to a combination of building blocks, which is indeed a procedure that is often followed by novice graffiti artists when learning how to sketch the stylised outlines a piece (Figure 12.3.a and b). However, with experience, certain combinations of strokes are assimilated, and the resulting letter outlines are often sketched as single entity (Figure 12.3.c). This results in a more rapid sketching procedure, in more organic letterforms, and in a drawing procedure that is akin to improvisation. Each new stroke is influenced by the previous one, or also by errors occurring while performing drawing gestures. This results in a feedback mechanism that makes it difficult to predict what the final drawing outcome will be. At the same time, designing graffiti with such a procedure requires first having a solid grasp of a set of component elements, and the primitives developed in this thesis are intended to provide such a basis to be used for more sophisticated models of the graffiti sketching process to come.

I have demonstrated in Chapter 11 how a stroke-based representation of a glyph, augmented with connectivity information, can be exploited to apply structural stylisations to a letterform. In Chapter 5, I also demonstrated simple examples where motor plans are concatenated to produce smooth ligatures. These are just demonstrative examples of the flexibility of the combined motor plan and stroke representations. Possible extensions to these simple procedures include: computing connections between strokes, replacing spine corners with loops or physics-based rigid deformations of a motor plan. The compositional and form functions proposed by Ferri (2016), which I have translated from Italian in Appendix

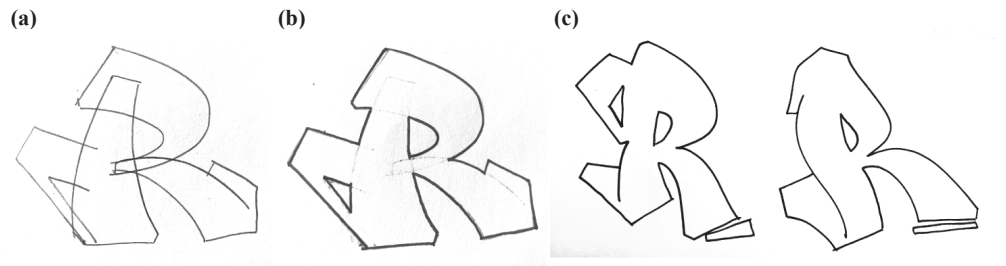


Figure 12.3: Sketching graffiti letters by hand. **(a)** sketching single strokes and then **(b)** outlining the union of the strokes. This procedure is usually followed when learning to draw graffiti. **(c)** With experience, the strokes are conceptualised and the letter outline is sketched as a whole.

B, are a useful guide for these future developments. This thesis implements computational analogues of a subset of these functions, but many more remain to be explored. The reader is invited to refer to Appendix B for an overview of functions that can be implemented as an extension to the methods presented in this thesis.

12.4.4 Empirical aesthetics research

The work developed in this thesis has contributed to a series of ongoing experiments that investigate the aesthetics of graffiti and the relations between the kinematic qualities of a movement and the aesthetic appreciation of the resulting drawing outcomes. The experiments (Chamberlain et al., 2020, 2019) have been designed and conducted by a team of psychologists at Goldsmiths, University of London and the Katholieke Universiteit in Leuven, with my contribution being the selection and generation of stimuli.

12.4.4.1 Perception of graffiti compared to text-based and pictorial art

In a first experiment (Chamberlain et al., 2020), we examined low level image statistics (self-similarity, anisotropy and complexity) and aesthetic ratings of graffiti art (pieces and tags) compared to other text-based (calligraphy, initiums and ornate lettering) and pictorial (abstract and representational painting) art forms. The experiment was conducted on 169 participants with varying degree of expertise in the arts. Participants showed a generally lower preference for graffiti art, when compared to other art forms. However the results also showed a preference for graffiti and calligraphy in expert over non-expert viewers, suggesting a role of expertise in the aesthetic appreciation of these art forms. Participants also preferred graffiti images with higher self-similarity and complexity, and lower anisotropy. This suggests that low level image statistics are an interesting metric to consider if integrating the methods described in this thesis into a procedural generation pipeline.

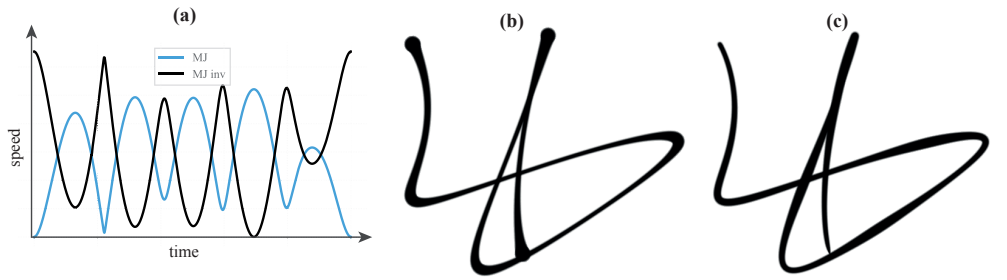


Figure 12.4: Example of minimum jerk and inverted speed profiles together with the corresponding static stimuli. The speed profiles are different, but the trace is the same. **(a)** Minimum jerk (MJ, black) and inverted minimum jerk (IMJ, blue) speed profiles. **(b)** Brush rendering of the MJ trajectory resulting in a thinner stroke near curvature extrema. **(c)** Brush rendering of the IMJ trajectory, resulting in a thinner stroke near curvature extrema.

12.4.4.2 Movement kinematics and the aesthetic appreciation of tags

In a second experiment (Chamberlain et al., 2019) we shifted our focus to synthetic tag stimuli, investigating the role of movement qualities in the aesthetic appreciation of tags. The experiment was conducted on 61 participants, of which 29 "experts" had at least 3 years of prior training in the arts. The experiment evaluated the perceived *naturalness* of biologically *feasible* and *infeasible* motions, and compared the naturalness ratings to aesthetic ratings of the resulting *static* traces. The biologically feasible motions followed minimum jerk trajectories, characterised by the stereotypical inverse relation between speed and absolute curvature. The trajectories were first generated with MIC using the method described in Section 5.3.3. We then reconstructed the trajectories using the path-constrained minimum jerk model (Todorov and Jordan, 1998), given its stronger biological implications with respect to MIC. The biologically infeasible motions followed an "inverted" reparameterisation of the same trace (Figure 12.4a), characterised by an opposite and implausible relation between speed and absolute curvature (Dayan et al., 2007). Both stimulus types were rendered with the brush model described in Section 4.5, resulting in thicker strokes near curvature extrema for the minimum jerk trajectories (Figure 12.4b) and the opposite relation for the reparameterised trajectories (Figure 12.4c).

In a first block of trials, the participants were asked to aesthetically rate static images, consisting of traces rendered with either the minimum jerk or inverse parameterisation. In a second block of trials, the participants were asked to rate the naturalness of the corresponding movements. All participants rated the minimum jerk trajectories as more natural and more aesthetically pleasing, with a strong correlation between naturalness and aesthetic ratings. Both expert and non expert participants found the movements generated with the minimum jerk model more natural than those generated with the inverse parameterisation, fur-

ther showing a strong correlation between the perceived naturalness of a movement and the aesthetic appreciation of the corresponding static stimulus. More surprisingly, only experts viewers showed a preference for the static stimuli generated with minimum jerk model.

Again, these results demonstrate a role of expertise in the aesthetic evaluation of graffiti. The results also support an embodied aesthetics hypothesis (Freyd, 1983; Freedberg and Gallese, 2007; Pignocchi, 2010) and further suggest that expert observers take into account the kinematic feasibility of a movement when aesthetically evaluating the resulting static trace, also for the case of synthetically generated drawings. In a follow-up series of experiments, we plan to construct stimuli from the digitised movements of a number of expert graffiti artists. We have already collected data from three artists by recording their writing movements with a whiteboard-marker digitiser.[2] We digitised different instances of tags, as well as random scribbles and isolated letters of the alphabet. We plan to use these movements as stimuli in the context of a series of EEG studies, in order to investigate possible neural correlates of the perceived naturalness and aesthetic quality of the digitised movements, as well as their reconstruction with the $\Sigma\Lambda$ and minimum jerk models.

12.4.5 Parameter choices and evaluation

Many of the parameter choices used across the thesis are driven by my personal aesthetic preference and experience as a graffiti writer. In particular, when using MIC, I have often settled for a relatively high system order of 4 (snap) or 5 (crackle) instead of the more conventional order of 3, which would be consistent with the minimum jerk model. This choice is based on my observation that these higher orders system result in subtle trajectory qualities that resemble ones that can be seen in well executed tags. Similarly, a user of the system is able to tune these parameters to his/her aesthetic preference. However, I argue that an experimental evaluation of the aesthetic appeal of different system orders is not only interesting in the context of graffiti modeling but also in the broader context of computational motor control, where the subject of "optimality" is still a matter of debate (Flash and Hogan, 1985; Edelman and Flash, 1987; Dingwell et al., 2004; Djioua and Plamondon, 2010).

Similarly, the choice of the CSF saliency measure in Chapter 7 has been determined based on its favourable performance in the applications discussed in Chapter 8 and Chapter 10. At the same time, the results of the large-scale experiment performed by De Winter and Wagemans (2008b) are publicly available, so in future studies I plan to compute correlations between the proposed saliency measure with the saliency computed from human preferences and then compare the results with the other measures studied by the authors of the study (e.g. turning angle and stick-out).

[2] The "eBeam smart marker" `https://www.luidia.com/smartmarker/`

12.4.6 Data driven methods

I have argued in Chapter 11 against the utility of controlled user studies to evaluate the aesthetic quality of results. The system is intended to give sufficient exploratory freedom to users, the aesthetic preferences of whom may vary depending on culture or simply personal taste (Hertzmann, 2010). However flexible, the large number of parameters of the system can be daunting to control and, often, settings that work well for one font are not guaranteed to work as well for another font. One interesting avenue of future research is to use a supervised learning method similar to the one used by Xu et al. (2012) for calligraphy to automatically evaluate stylisation results based on the feedback of one or more expert graffiti artists. Another interesting approach would be to combine a form of dimensionality reduction (Yumer et al., 2015) together with a genetic algorithm (McCormack and Lomas, 2020) to reduce the number of parameters exposed to a user and to automatically predict settings given user selected preferences.

A data driven approach is also potentially useful to efficiently solve the previously discussed parameterisation problem with MIC. Optimal passage times can be computed with an optimisation method similar to the one proposed by Todorov and Jordan (1998) for the minimum jerk model. This optimisation results in curvature extrema that occur at motor plan vertices for interpolatory trajectories or near these vertices for approximating ones. However, this kind of optimisation problem cannot currently be solved at interactive rates. One potential interactive solution consists in computing optimal passage times offline for a number of motor plan configurations, and then learning a mapping between relative orientations of motor plan segments to the optimal passage times with a sequence-based model similar to those discussed in Chapter 9. Conceptually similar approaches have become recently popular in fluid or smoke simulations (Wiewel et al., 2019; Kim et al., 2019).

Finally, data driven methods are also a promising extension to the junction classification procedure in Chapter 10. We currently rely on a heuristic method, with parameters determined empirically based on a quantitative and qualitative evaluation of the results. In future developments the same choices could be driven with examples labelled by a user or automatically and the junction representation is sufficiently high-level to hypothesise that only a few examples should be sufficient for this task. As mentioned in Chapter 10, the current segmentation method already produces paths that can be used to train sequence-based generative models, transforming font outlines into potential training data for methods such as the one developed by Ha and Eck (2018) or Lake et al. (2013). Investigating the performance of these methods to generate novel letter structures is also an interesting and promising avenue of future research.

12.5 Final notes

Returning to Hofstadter (1985), it should be noted that the system developed in this thesis does not attempt to model the creative process involved in graffiti creation, but rather to provide a "creativity support tool" (Shneiderman, 2009; Resnick et al., 2005), which can enable a more expert user to rapidly prototype complex graffiti-like compositions and a more novice user to rapidly create stylised strings that resemble this art form. This is not to say that modeling higher level aspects of the graffiti production and conception process is not a useful future research avenue. The primitives developed in this thesis are meant to provide an abstraction layer that facilitates graffiti content creation for a human, but also for an AI system that would be left with a simplified and higher level task of planning and composition.

This thesis is concerned with the stylistic and visual aspects of graffiti art. However, there is much more to this art form, which also includes a rich subculture, made of people around the globe sharing an unusual artistic interest and often adventurous experiences, which certainly cannot be replaced with a simulation, let alone with the methods developed here. At the same time, as a graffiti artist, I hope that the tools I have developed can contribute to the growing adoption of technology in graffiti art, and that results produced using these tools will eventually find a place as physical graffiti (more or less automatically) made in the (urban) wild.

Appendix A

List of peer-reviewed publications

The following is a chronological list of publications that have been peer-reviewed, accepted and published, and for which I am the first author; followed by a reference to the Chapters where they are most relevant.

1. (Berio and Leymarie, 2015) D. Berio and F. F. Leymarie. "Computational models for the analysis and synthesis of graffiti tag strokes." In P. Rosin, editor, *Computational Aesthetics*, pages 35–47. Eurographics Association. Proceedings of a workshop part of the Expressive Symposium, held in Istanbul, Turkey, June 2015. (Chapters 4 & 8).

2. (Berio et al., 2016) D. Berio, S. Calinon, and F. F. Leymarie. "Learning dynamic graffiti strokes with a compliant robot." In the Proceedings of the *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3981–3986. Held in Daejeon, South Korea, October 2016. (Chapter 4).

3. (Berio et al., 2017b) D. Berio, S. Calinon, and F. Fol Leymarie. "Generating calligraphic trajectories with model predictive control." In *Proceedings of Graphics Interface*, Canadian Human-Computer Communications Society. Held in Edmonton, Canada, May 2017. (Chapter 5).

4. (Berio et al., 2017d) D. Berio, F. Fol Leymarie, and R. Plamondon. "Computer aided design of handwriting trajectories with the kinematic theory of rapid human movements." In the Proceedings of the *18th Biennial Conference of the International Graphonomics Society* (IGS). Held in Gaeta, Italy, June 2017. Received the GIRPR **best paper award** on Pattern Recognition, sponsored by Gruppo Italiano Ricercatori in Pattern Recognition. (Chapter 4).

5. (Berio et al., 2017c) D. Berio, S. Calinon, and F. F. Leymarie. "Dynamic graffiti stylisation with stochastic optimal control." In the ACM *Proceedings of the 4th International Con-*

ference on Movement Computing (MOCO). Held in London, UK, June 2017. (Chapter 5).

6. (Berio et al., 2017a) D. Berio, M. Akten, F. F. Leymarie, M. Grierson, and R. Plamondon. "Calligraphic stylisation learning with a physiologically plausible model of movement and recurrent neural networks." In the ACM *Proc. of MOCO*. Held in London, UK, June 2017. (Chapter 9).

7. (Berio et al., 2018a) D. Berio, F. F. Leymarie, and R. Plamondon. "Expressive curve editing with the sigma lognormal model. " In the *Proceedings of the 39th Annual European Association for Computer Graphics Conference: Short Papers*, pp. 33–36. Eurographics. Held in Delft, the Netherlands, April 2018. (Chapter 4).

8. (Berio et al., 2018b) D. Berio, F. F. Leymarie, and R. Plamondon. "Kinematic reconstruction of calligraphic traces from shape features." In the *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence* (ICPRAI), vol. 1, pp. 762–767. Held in Montreal, Canada, in May 2018. (Chapters 7 & 8).

9. (Berio et al., 2019) D. Berio, P. Asente, J. Echevarria, and F. F. Leymarie. "Sketching and layering graffiti primitives." In the Proceedings of the *8th ACM/Eurographics Expressive Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pp. 51–59. Held in Genoa, Italy, May 2019. (Chapter 6).

10. (Berio et al., 2020a) D. Berio, F. F. Leymarie, and S. Calinon. "Interactive generation of calligraphic trajectories from Gaussian mixtures." In *Mixture Models and Applications*, Unsupervised and Semi-Supervised Learning book series (UNSESUL), Springer, Ch. 2, pp. 23–38. 2020 (Chapter 5).

11. (Berio et al., 2020b) D. Berio, F. F. Leymarie, and R. Plamondon. "Kinematics reconstruction of static calligraphic traces from curvilinear shape features." In *The Lognormality Principle and its Applications in e-Security, e-Learning and e-Health*, Series in Machine Perception and Artificial Intelligence. Ch. 11, pp. 237–268. World Scientific. December 2020. (Chapters 7 & 8).

Appendix B

Ferri's form and composition functions

The following are translations from Italian, with annotations, of two tables from the book of (Ferri, 2016) "Teoria del writing, La ricerca dello stile". Ferri describes a number of "compositional" (Table B.1) and "form" (Table B.2) operations, that according to him characterise different graffiti styles ("style" columns, *cf.* Figure 1.5 in the intro). The operations are not defined computationally or mathematically, but many of these afford a computational interpretation. A subset of these operations are, at least partially, implemented in this thesis. These are color coded (red or blue) according to the relevant chapter in the same row. I also indicate that a subset of the remaining operations can readily be implemented in the near future, and I color code them as future work.

FC#	Style	Description	Operations	Implementation
FC0	stick	Outlining a shape or area in order to create a form	Adjacent; Spaced; Overlapped/(underlapped); crossing/intersections; loops; knots; serifs;	Chapter 6, Chapter 11
FC1	bubble	Addition/union	Adjacent; Spaced; Overlapped; crossing/intersections; Fused; Outlined; Incorporation;	Chapter 6, Chapter 11
FC2	bubble	Subtraction	Subtraction of a overlapped/intersecting shape from another; "Partial" subtraction of a shape; Complete subtraction of a shape from another; Inverse subtraction (XOR).	Future work.
FC3	bubble,block	Intuitive/rational repetition	adjacent; spaced; overlapped; Crossing/intersections; Fused; Increasing/decreasing; ordered; random; tilted; "intuitive" or systematic	Chapter 6.
FC4	block	Subdivision of space	adjacent; spaced; overlapped; crossing/intersections; fused; increasing/decreasing; ordered; random; tilted; subdivision of space;	Chapter 6
FC5	marshmallow	Combined application of bubble functions, to stick or block.	Deform; Distort; symbolic/figurative sculpting; application to form of morphological/natural attributes.	Future work.
FC6	platform	Application of block functions, to stick or bubble, through structural forms.	Platforms; Connections; Structure; intertwinement of structures and schemas.	Future work
FC7	combo	Combined application of bubble, stick and block functions through structural schemas.	Combine; progression; skeme/trace; structure/skeleton; rhythmic/metric; tangle;	Chapter 11, Chapter 10, Future work.
FC8	arrow	Multiplication between different forms. Combined application of stick, bubble, block, marshmallow, platform and combo functions.	Compositional method: scheme/diagram, structure, trace; Movement; Interior, exterior and outline have the same importance; Branching; Explosion; Expansion; Multiplication between different forms;	Chapter 11, Chapter 5, Future work.
FC9	puzzle	Division between different forms. Combined application of stick, bubble, block, marshmallow, platform and combo functions.	Deconstruction/decomposition; Division between forms; External influences; Patterns; Abstraction; Informal; Interior, exterior and outline have the same importance;	Future work.
FC10	machine	Equations between forms	Adding architectural or mechanical attributes to forms; Citations, mixing, influences; Interior, exterior and outline have the same importance;	
FC11	wild style	Free/improvised combination of functions, which determine the individual's style.	infinite	

Table B.1: Compositional functions

F#	Style	Description	Operations	Implementation methods
F0	stick	Force of the "sign" (movement). Determines basic shapes of Sticks and Bars.	Line; Angles/curves; Parabola; broken, acute, obtuse; Mixed; Start and end of shape, extremities, serifs; Open/closed shape	Chapters 5 and 6
F1	stick	Force that is "internal" to the shape	Length/width; Increase area in one or both directions	Chapter 6
F2	stick	Force that is "external" to the shape	Width; Decrease/shrink area non proportionally.	Chapter 6
F3	stick	Folding	Folds the sign, creates overlaps; loops	Chapter 6
F4	bubble	Intuitive study (speculation) of form. "Soft" force.	Fattening; Magnification; Inflation; Modelling; "moderate" torsion	Future work.
F5	bubble	Shift	translation	
F6	bubble	Intuitive 3d effects	Shadows, thickness	Chapter 6
F7	bubble, block	Application of geometric study of shape. Intuitive/rational.	Section/division; tilt; rotation; curvature; schematisation; projection	Chapter 6, Chapter 11, Future work.
F8	block	"rigid" three-dimensionality	Volume; XYZ Projection; XYZ fold; Perspective; Relief;	Chapter 6, Future work.
F9	marshmallow	Figurative/symbolic modelling	shaping/modelling; distortion; deformation; figurative/organic attributes.	Chapter 6, Future work.
F10	marshmallow	Organic reactions, metamorphosis	cracks; breakages; holes; deflations; melting; drips; loss of outline; bouncing.	Future work.
F11	platform	Structural function of elements	arrangement; base (platform); roof; support; connections; loops.	Future work.
F12	combo	Synergetic combination of form functions and compositional functions for stick, bubble, block, platform and marshmallow	The form is given by simple combinations of elements with semantic attributes (the word in language).	Future work.
F13	arrow	Form multiplication functions	Multiply; Motion, dynamics; Interior, exterior and outline have the same importance; simple semantic combination of elements (language).	Chapter 6.
F14	puzzle	Form division functions	Divide; Abstraction; Interior, exterior and outline have the same importance. Simple semantic combination of elements (word, language).	Future work.
F15	machine	Form results from complex combinations of form functions and compositional functions for combo puzzle arrow and all other styles.	Form emerges from the combination of form functions and compositional functions; Attribution of semantic and significant values to combination of forms. Equivalent to a concept expressed with a sentence.	

Table B.2: Form functions

Appendix C

Additional details on MIC trajectory generation

C.1 Displacement-based smoothing weight

Several advanced methods exist to automatically tune the tracking and control weights for LQT problems, but often the weights are chosen with a trial and error procedure. A well known method to choose weights is known as Bryson's rule (Hespanha, 2005), in which the values along the diagonals of $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ are chosen as

$$\frac{1}{{e_{\max}}^2} \quad \text{and} \quad \frac{1}{{u_{\max}}^2}$$

where $e_{\max}$ and $u_{\max}$ are respectively the maximum desired state deviation and command magnitude. Bryson's rule produces a dimensionless cost function, since the units of the squared denominators and the quadratic terms in Eqn. 5.5 cancel out.

Another weight tuning method is based on the observation that plotting the magnitude of the generated commands against the log of the residual in the least squares estimate as a function of a set of constant values along the diagonal of $\boldsymbol{R}_t$ results in an L-shaped curve (Hansen, 2000; Zeestraten et al., 2016a). The point of maximum curvature in the L-curve is then considered an optimal choice for the weights in $\boldsymbol{R}_t$.

With the problem at hand, we are interested in allowing a user to rapidly explore the visual quality of a trajectory across different system orders, and the hand-application of Bryson's rule is not practical because it is is necessary to estimate the maximum forcing command for high order derivatives of position. At the same time, the L-curve method requires multiple iterations per order to find the optimal value, which is not practical in our use case for obvious performance reasons.

In order to automatically adjust the weights in $\boldsymbol{R}$ in a manner that is independent from

the order of the system, we can use the transfer function of the integrator chain, which for the continuous system case is simply given by

$$H(s) = \frac{\mathscr{L}[y(t)]}{\mathscr{L}[u(t)]} = \frac{Y(s)}{U(s)} = \frac{1}{s^n} \quad , \tag{C.1}$$

where $\mathscr{L}$ is the Laplace transform operator.

We then compute the gain of the system at a low frequency, which we empirically choose by using a time period $T_s\Delta t$, resulting in a natural frequency of

$$\omega = \frac{2\pi}{T_s\Delta t}. \tag{C.2}$$

In practice this corresponds to the assumption that a trajectory is decomposed into a discrete sequence of ballistic submovements aimed at consecutive targets, and we are measuring the gain of an oscillatory motion between two targets, with a period given by the average duration of a sub-movement.

We then express the weights $\boldsymbol{R}$ is then expressed in terms of maximum *displacement* $\Delta_{\max}$ rather than an order dependent command amplitude, with

$$\boldsymbol{R} = \left(\frac{\left|H(\omega j)\right|}{\Delta_{\max}}\right)^2 \boldsymbol{I} = \frac{1}{(\omega^n\Delta_{\max})^2}\boldsymbol{I} \quad , \tag{C.3}$$

where ωj is the complex frequency and $\Delta_{\max}$ is the maximum displacement. If we examine the dimensionality of the terms in the cost function, we see that the dimension of $[\boldsymbol{R}_t]$ is $[\frac{T^{2n}}{L^2}]$ which results in a dimensionless term $\boldsymbol{u}^\top\boldsymbol{R}_t\boldsymbol{u}$ in 5.5. The Manhanolobis distance in the state term is dimensionless as well, therefore the whole cost function is dimensionally consistent.

The choice of the period 2π in the numerator of Eqn. C.2 produces consistent tracking results across different system orders, but it makes a precise assumption on the structure of a movement. Intuitively, another reasonable choice for the numerator would be π instead of 2π. This would correspond to measuring the gain of one point-to-point ballistic movement, but this choice did not produce satisfactory results in our experiments.

C.1.1 Derivation with Simple Harmonic Motion

An equivalent result can be derived by considering an idealised oscillatory motion (Simple Harmonic Motion) between consecutive targets with equation

$$\tilde{x}(t) = \Delta_{\max}\cos(\omega t) \quad ,$$

where $\Delta_{\max}$ is the amplitude (maximum displacement) of the motion and ω is the angular frequency of the oscillation. The absolute value of the n^{th} order derivative of $\tilde{x}$ is given by

$$\left|\frac{d^n\tilde{x}}{dt^n}\right| = \begin{cases} \omega^n\Delta_{\max}\sin(\omega t) & \text{if } n \text{ is odd,} \\ \omega^n\Delta_{\max}\cos(\omega t) & \text{if } n \text{ is even.} \end{cases}$$

and has a maximum amplitude given by $\omega^n\Delta_{\max}$, which we can use to construct the diagonal of $\boldsymbol{R}_t$ as a function of the user specified displacement $\Delta_{\max}$ with

$$\boldsymbol{R}_t = \frac{1}{(\omega^n\Delta_{\max})^2}\boldsymbol{I} \qquad \text{and} \qquad \omega = \frac{2\pi}{T_s\Delta t} \quad , \tag{C.4}$$

where ω is empirically set to the period corresponding to the average sub-movement duration $T_s\Delta t$, and the denominator is squared because the control term $\boldsymbol{u}_t^\top\boldsymbol{R}_t\boldsymbol{u}_t$ in Eqn. 5.5 is quadratic. Again, if we examine the dimensionality of the terms, we see that the dimensions of $[\boldsymbol{R}_t]$ and $\boldsymbol{u}_t^\top\boldsymbol{u}_t$ are respectively $[\frac{T^{2n}}{L^2}]$ and $[\frac{L^2}{T^{2n}}]$, which cancel out and result in a dimensionless cost function.

C.2 Iterative solution

A more efficient solution to the discrete tracking problem can be derived using dynamic programming, with a technique that is often the basis for control problems. However, this has the disadvantage of not producing an output trajectory distribution or allowing the trivial generation of periodic motions. We refer the interested reader to the work of Bryson (Bryson, 1999) for the details of the derivations. It follows that the optimal solution is given in the form of a feedback controller with time-varying weighting matrix $\boldsymbol{K}_t$, and the commands for each time step t are given by

$$\boldsymbol{u}_t = \underbrace{-\left(\tilde{\boldsymbol{B}}^\top\boldsymbol{P}_t\tilde{\boldsymbol{B}} + \boldsymbol{R}_t\right)^{-1}\tilde{\boldsymbol{B}}^\top\boldsymbol{P}\tilde{\boldsymbol{A}}}_{\boldsymbol{K}_t}\tilde{\boldsymbol{x}}_t, \tag{C.5}$$

where

$$\boldsymbol{P}_t = \tilde{\boldsymbol{Q}}_t - \boldsymbol{A}^\top\left(\boldsymbol{P}_{t+1}\tilde{\boldsymbol{B}}\left(\tilde{\boldsymbol{B}}^\top\boldsymbol{P}_{t+1}\tilde{\boldsymbol{B}} + \boldsymbol{R}_t\right)^{-1}\tilde{\boldsymbol{B}}^\top\boldsymbol{P}_{t+1} - \boldsymbol{P}_{t+1}\right)\tilde{\boldsymbol{A}} \tag{C.6}$$

is a *Riccati difference equation*, which can be solved backwards in time by setting a terminal condition $\boldsymbol{P}_N = \tilde{\boldsymbol{Q}}_N$. In equations (C.5) and (C.6), we introduce the symbols $\tilde{\boldsymbol{x}}_t$, $\tilde{\boldsymbol{Q}}_t$, $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$. These respectively denote an *augmented* state vector and tracking weight

$$\tilde{\boldsymbol{x}}_t = \left[\bar{\boldsymbol{x}}_t^\top, 1\right]^\top \quad \text{and} \quad \tilde{\boldsymbol{Q}}_t = \begin{bmatrix} \boldsymbol{Q}_t^{-1} + \bar{\boldsymbol{x}}_t\bar{\boldsymbol{x}}_t^\top & \bar{\boldsymbol{x}}_t \\ \bar{\boldsymbol{x}}_t^\top & 1 \end{bmatrix}^{-1} , \tag{C.7}$$

and augmented system matrices

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix}. \tag{C.8}$$

This allows the tracking problem to be treated more compactly and efficiently as a *regulation* problem, resulting in a formulation that is equivalent to a Linear Quadratic Regulator (LQR).

Appendix D

Additional details for font segmentation

D.1 Association fields

Our association fields are adapted from Ernst et al. (2012). The model predicts the conditional link probability of one oriented element relative to another one. The link probability α is given by the product $A^{\phi}A^{d}$ of an angular and a radial component. The angular component parameterises deviations from perfect cocircularity and deviations from zero curvature with the product of two von Mises distributions, analogs of Gaussian distributions with a circular support. Given two orientations ϕ_i, ϕ_j and planar positions $(x_i, y_i), (x_j, y_j)$ the angular component simplifies to:

$$A^{\phi} = \frac{C}{4}\cosh\left(\frac{1}{\sigma_{\beta}^{2}}\cos\left(\beta/2\right) + \frac{1}{\sigma_{\theta}^{2}}\cos\left(\theta-\beta/2\right)\right) \quad , \tag{D.1}$$

with $\beta = \phi_j - \phi_j$, $\theta = \tan^{-1}\left(\left(y_j - y_i\right)/\left(x_j - x_i\right)\right) - \phi_i$, and $\sigma_{\theta} = 0.27$ and $\sigma_{\beta} = 0.47$ spread parameters for cocircularity and curvature respectively.[1] We use the values for the spread parameters that were experimentally found to be optimal by Ernst et al. (2012). The constant C is a normalization factor derived from the von Mises distribution with:

$$C = \pi^2 I_0\left(1/\sigma_a^2\right) I_0\left(1/\sigma_b^2\right) \quad , \tag{D.2}$$

where I_0 is the modified-Bessel function of the first kind with order 0. We also divide A^{ϕ} by 0.602, so it falls in the $[0,1]$ range, which facilitates parameter setting in our application-driven use case.

For the task of grouping closely-spaced oriented elements, Ernst et al. (2012) express the radial component as an exponential function that decays with distance. Again, we opt for a formulation that facilitates parameter tuning and express the component with a Gaussian

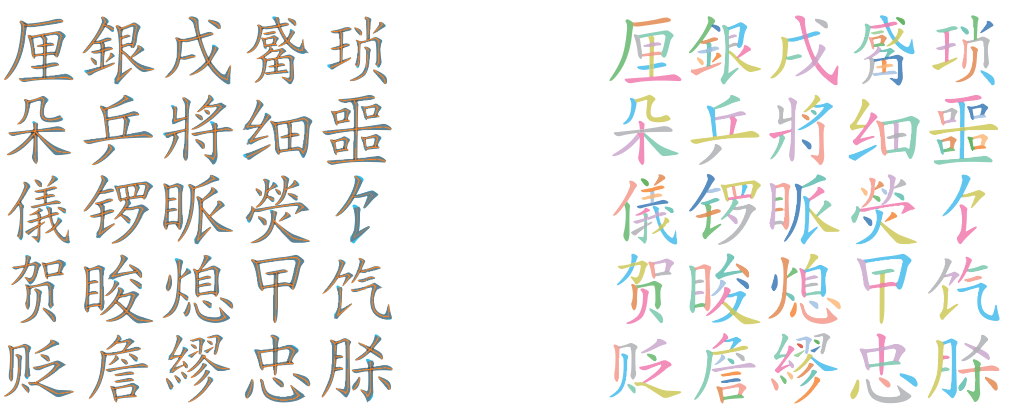
[1] This equation corrects a typographic error in the original paper

decay:

$$A^d = \exp\left(\frac{d^2}{2\sigma_d^2}\right) \quad , \tag{D.3}$$

with d the distance between the two positions and σ_d a distance-spread. We set σ_d to twice the maximum SA_+^I radius when computing good continuation for splits, and to the distance between the branch tangent origins during Y-junction interpretation.

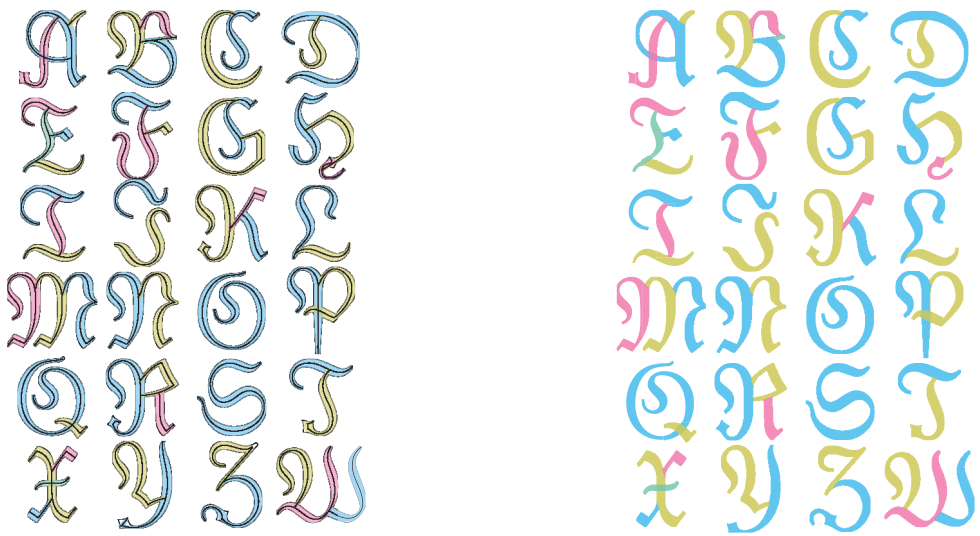
D.2 Hanzi segmentation examples



(a) Paths and width profiles **(b)** Areas

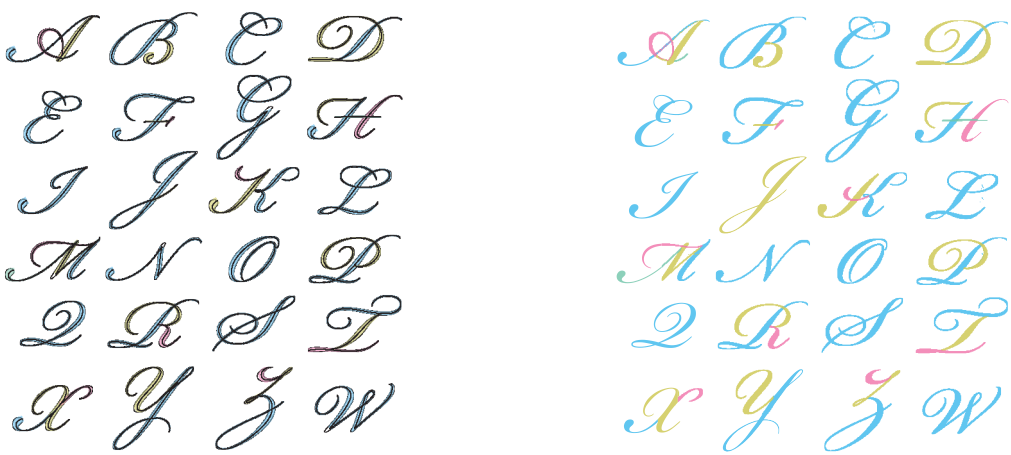
Figure D.1: Example segmentations from the *make-me-a-hanzi* dataset (Kishore, 2018)

D.3 Font segmentation examples



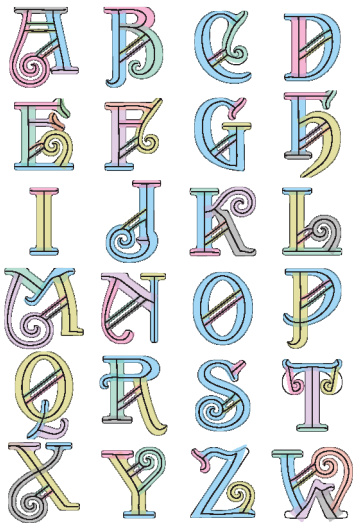
(a) Paths and width profiles **(b)** Areas

Figure D.2: Font: Moderne Fraktur.

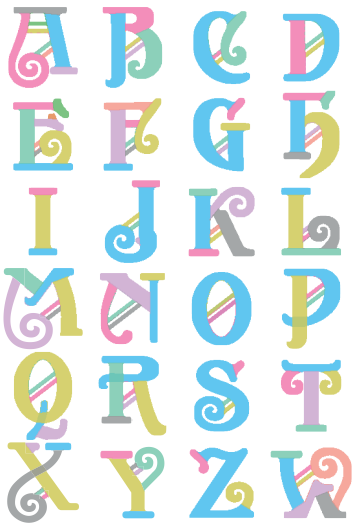


(a) Paths and width profiles **(b)** Areas

Figure D.3: Font: Bickham Script.



(a) Paths and width profiles



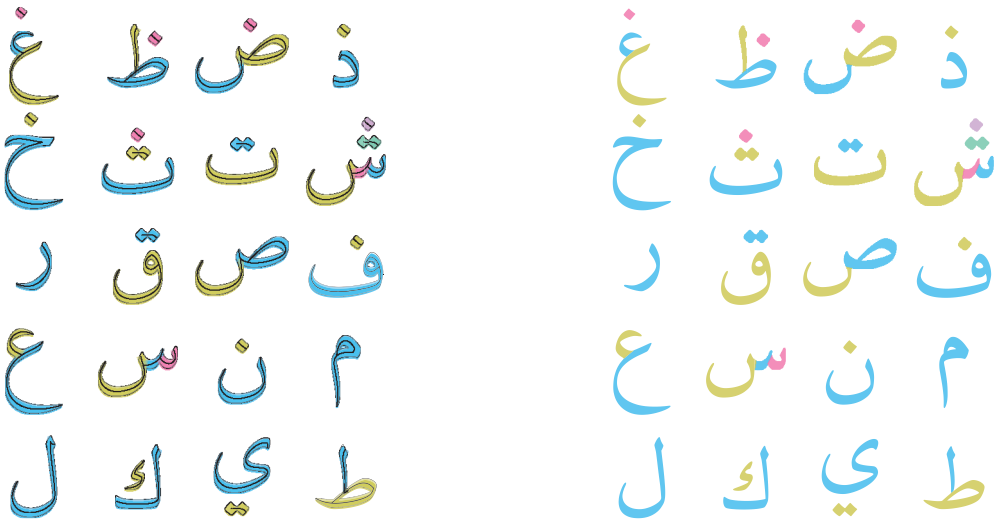
(b) Areas

Figure D.4: Font: Apollo.

(a) Paths and width profiles

(b) Areas

Figure D.5: Font: Arial bold.

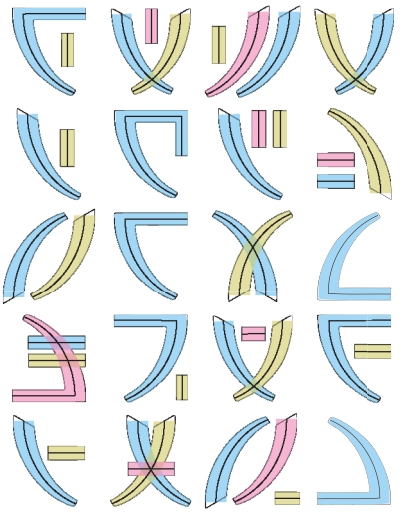


(a) Paths and width profiles **(b)** Areas

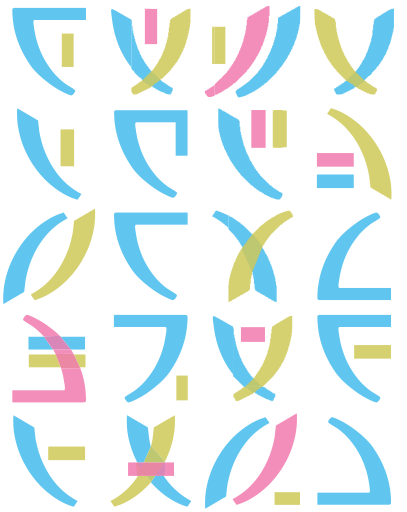
Figure D.6: Font: Adobe Arabic bold.

(a) Paths and width profiles **(b)** Areas

Figure D.7: Font: Adobe Hebrew bold.



(a) Paths and width profiles



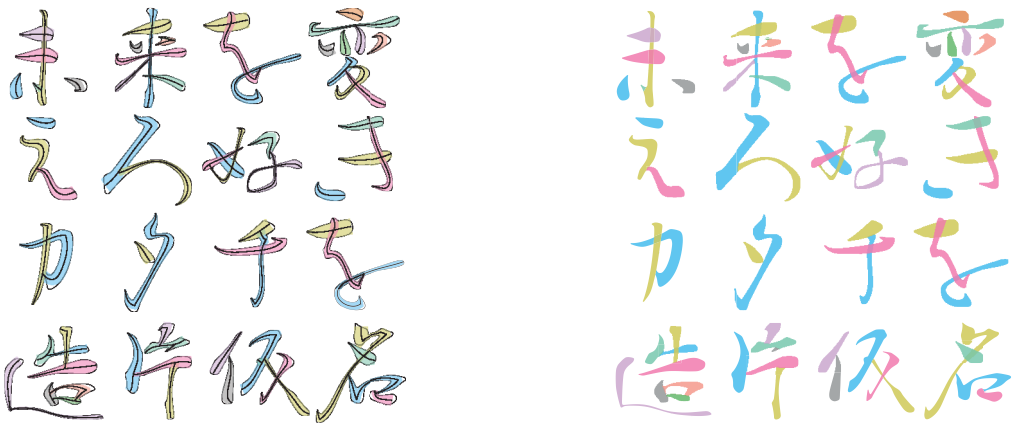
(b) Areas

Figure D.8: Font: PACL.

(a) Paths and width profiles

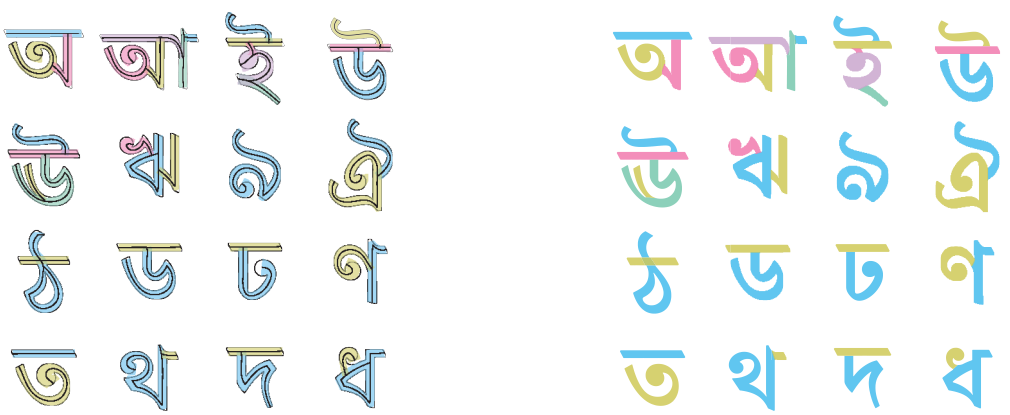
(b) Areas

Figure D.9: Font: Georgia.



(a) Paths and width profiles **(b)** Areas

Figure D.10: Font: Kazuraki.



(a) Paths and width profiles **(b)** Areas

Figure D.11: Font: Adobe Bengali Bold.

Appendix E

Symbols and values

The following is a list of the main symbols (Sections E.1 and E.2), functions (Section E.3), parameters (Section E.4) and thresholds (Section E.5) used across the thesis. Sections E.4 and E.5 also specify default parameter and threshold value, unless the value is assumed to depend on a choice made interactively by a user, in which case the corresponding field is left empty.

E.1 Symbols (general):

Symbol	Description
$\boldsymbol{x}(t)$	Point along a trajectory evaluated at time t.
$\boldsymbol{x}_t$	Point along a trajectory evaluated at discrete time step t
$\boldsymbol{p}_i$	Position of a motor plan vertex.
$\boldsymbol{z}(s)$	Trace or contour $\boldsymbol{z}$, parameterised by arc length s.
$\boldsymbol{b}$	Bisector.
$\boldsymbol{t}$	Tangent.
κ	Curvature.
r	radius of curvature $\lvert 1/\kappa \rvert$.
$C(s), S(s)$	Cosine and sine Fresnel integrals.
$\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$	Multivariate normal distribution, with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Symbol	Description
$\mathcal{N}(\mu,\sigma)$	Univariate normal distribution, with mean μ and standard deviation σ.
P or Q	Motor plan.
$\mathscr{P}$ or $\mathscr{Q}$	Kinematic realisation of a motor plan (resp. P or Q).
Θ	Kinematic parameters.

E.2 Other symbols and objects:

Symbol	Description	Section
t_{0i}	Lognormal activation time.	§4.1
μ_i	Lognormal delay.	§4.1
σ_i	Lognormal response time.	§4.1
D_i	Lognormal amplitude.	§4.1
θ_i	$\Sigma\Lambda$ sub-movement orientation.	§4.1
δ_i	$\Sigma\Lambda$ model sub-movement curvature parameter.	§4.1
Δt_i	Lognormal time overlap parameter.	§4.2.3
A_{c_i}	Lognormal shape parameter.	§4.2.3
T_i	Lognormal duration parameter.	§4.2.3
s_{0i}, s_{1i}	$\omega\mathscr{E}\Sigma\Lambda$ sub-movement spiral parameters.	§4.2.2
τ_i	Time occurrence (or passage time) of *key-point* along a trajectory.	§4.4.2.1
$\bar{\boldsymbol{x}}_t$	MIC reference trajectory at time step t.	§5.1.3
$\boldsymbol{Q}_t$	MIC reference weights at time step t.	§5.1.3
$\boldsymbol{u}_t$	MIC control command at time step t.	§5.1.1.1
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$	Discrete-time system matrices	§5.1.1.1
$\boldsymbol{\xi}_i$	Kinematic parameter prediction vector.	§9.1.4
$\boldsymbol{z}_i$	Point along outline	§10.2.1.1
$\boldsymbol{t}_i$	Tangent at the edge of a concave contact region.	§10.2.1.1

Symbol	Description	Section
$\boldsymbol{b}_i$	Bisector at a concavity.	§10.2.1.1
SA	Symmetry axis.	§7.2
SA^I	Interior symmetry axis.	§7.2
SA^E	Exterior symmetry axis	§7.2
SA^I_+	Interior curvilinear augmented symmetry axis (CASA).	§7.3.4
SA^E_+	Exterior CASA.	§7.3.4
SA_i	Local symmetry axis.	§7.3
CC_i	CSF contact circle.	§7.3
$\overrightarrow{CC}_i$	CSF contact region (circular arc).	§7.3
$\hat{\boldsymbol{z}}_i$	CSF curvature extremum locus.	§7.3
$\boldsymbol{z}_i^{lhs}(s)$	Left CSF support segment.	§7.3
$\boldsymbol{z}_i^{rhs}(s)$	Right CSF support segment	§7.3
N_{CSF}	Number of CSFs.	§7.3
GH	Split and concavity graph.	§10.3
GX	Crossing graph.	§10.4.3.1
$\tilde{Q}$	Stroke area planar map.	§10.5.2

E.3 Functions:

Symbol	Description	Section
$\Lambda(t)$	Lognormal evaluated at time t	§4.1
$\phi_i(t)$	$\Sigma\Lambda$ model curvilinear evolution.	§4.1
$\phi_{si}(t)$	$\omega\mathscr{E}\Sigma\Lambda$ model curvilinear evolution.	§4.2.2
$\phi_b(x)$	Brush "hat" function.	§4.5
$h_i(t)$	Gaussian activation function.	§5.1.3
$\beta(b,f)$	Saliency of a branch b extending from fork f	§10.2.1.5
$\beta_c(b,f)$	Saliency of a branch with respect to concavity c	§10.2.1.5
$w(c)$	Saliency of a CSF c	§7.3.3
$\omega(\eta)$	Saliency of a split η	§10.3.3

Symbol	Description	Section
$d(c,v)$	Influence of a concavity c on vertex v	§10.2.1.1
$\boldsymbol{\pi}(b,f)$	Protruding direction of branch b from fork f	§10.2.1.4
$\boldsymbol{\varphi}(c_i,c_j)$	Flow direction for two concavities	§10.2.2
$\alpha(c_i,c_j)$	Good-continuation value between two concavities	§10.2.2
$\alpha(\eta_i,\eta_j)$	Good-continuation value between two splits	§10.4.3

E.4 Parameters:

Symbol	Description	Value	Section
k_δ	$\Sigma\Lambda$ curvature exaggeration.		§4.4.2.2
$k_{\Delta t}$	$\Sigma\Lambda$ time overlap exaggeration.		§4.4.2.2
σ_h	MIC time interval parameter.		§5.1.3
$\Delta_{\max}$	MIC maximum displacement parameter.		§5.1.4
σ_α	Fold amount	$\pi/4$	§6.1
λ_p	Virtual target adjustment weight.	0.5	§4.4.2.1 and §8.2.3
λ_Δ	$\Sigma\Lambda$ time-overlap adjustment weight.	0.1	§8.2.3
λ_δ	$\Sigma\Lambda$ curvature adjustment weight.	0.1	§8.2.3
Δs	Trace/contour sampling distance.	1	§7.2.1
h_{ext}	Reference bounding box height.	150	§7.2.1
k_c	Concavity influence spread	0.5	§10.2.1.1
λ_s	Branch saliency stick-out steepness	0.5	§10.2.1.5
λ_η	Split saliency length weight	2	§10.3.3
λ_r	Proximity radius multiple	3	§10.3.1.1

E.5 Thresholds and Tolerances:

Symbol	Description	Value	Section
$s_{\min}$	Chord residual threshold.	0.5	§7.2.1
-	Minimum CSF saliency $w(c)$ (M^+,m^-).	1×10^{-3}	§7.3.3
-	Minimum CSF saliency $w(c)$ (M^-,m^+).	1×10^{-6}	§7.4
-	Transition segment subdivision threshold.	$(4\pi)/5$	§7.5.1.1
-	Degenerate inflection threshold.	0.2	§8.1.1

Symbol	Description	Value	Section
-	Branch saliency threshold $\beta(b, f)$	0.5	§10.2.1.5
-	Split good-continuation threshold	0.15	§10.4.3
-	Split pairing max angle	45°	§10.4.3
τ_M	Morphological junction threshold	0.2	§10.4.4.2
γ_T	T-junction detection threshold	0.2	§10.4.4.4
γ_B	Blunt-tip detection threshold	1	§10.4.4.4
-	Flexure minimum influence	0.77	§10.4.4.4
Υ_l	Relative root length threshold	3	§10.4.4.5
Υ_w	Width disparity threshold	1.3	§10.4.4.5
–	Degree of overlap δ_C threshold	0.98	§7.3.2
ϵ_θ	Local convexity angle tolerance	15°	§10.3.1.2

Bibliography

E. Abbena, S. Salamon, and A. Gray. *Modern differential geometry of curves and surfaces with Mathematica*. CRC press, 2017.

W. Abend, E. Bizzi, and P. Morasso. Human arm trajectory formation. *Brain: A journal of neurology*, 105 (Pt 2):331–348, 1982.

Adobe. Adobe animate: User guide, 2019a. URL `https://helpx.adobe.com/animate/user-guide.html`.

Adobe. Illustrator: User guide, 2019b. URL `https://helpx.adobe.com/illustrator/user-guide.html`.

L. Albertazzi. Styled morphogeometry. *Axiomathes*, pages 1–24, 2019.

L. Albertazzi, L. Canal, R. Micciolo, and M. Vescovi. Calligraphy and Klee's abstract painting: A study on categorical ambiguity. *Art & Perception*, 3(3):239—263, 2015.

Z. AlMeraj, B. Wyvill, T. Isenberg, A. A. Gooch, and R. Guy. Automatically mimicking unique hand-drawn pencil lines. *Computers & Graphics*, 33(4):496–508, 2009.

N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999.

X. Ao, Q. Fu, Z. Wu, X. Wang, M. Zhou, Q. Chen, and H. S. Seah. An intersection algorithm for disk B-spline curves. *Computers & Graphics*, 70:99–107, 2018.

P. Aparajeya and F. F. Leymarie. Point-based medialness for 2D shape description and identification. *Multimedia Tools and Applications*, 75:1667–1699, February 2016.

A. Appel. The notion of quantitative invisibility and the machine rendering of solids. In *Proceedings of the 1967 22nd National Conference*, ACM '67, pages 387–393. Association for Computing Machinery, 1967.

E. Arias-Castro, G. Lerman, and T. Zhang. Spectral clustering based on local PCA. *Journal of Machine Learning Research*, 18:1–57, 2017.

R. Arnheim. *Art and visual perception: A psychology of the creative eye*. Univ of California Press, 1954.

A. Arte. *Forms of Rockin': Graffiti Letters and Popular Culture*. Dokument Press, 2015.

H. Asada and M. Brady. The curvature primal sketch. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):2–14, January 1986.

P. Asente, M. Schuster, and T. Pettit. Dynamic planar map illustration. In *ACM Transactions on Graphics (TOG)*, volume 26, page 30. ACM, 2007.

P. J. Asente. Folding avoidance in skeletal strokes. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, pages 33–40. Eurographics Association, 2010.

F. Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183, 1954.

J. August, K. Siddiqi, and S. W. Zucker. Contour fragment grouping and shared, simple occluders. *Computer Vision and Image Understanding*, 76(2):146–162, 1999. ISSN 10773142.

V. Ayzenberg and S. F. Lourenco. Skeletal descriptions of shape provide unique perceptual information for object recognition. *Scientific reports*, 9(1):1–13, 2019.

S. Azadi, M. Fisher, V. G. Kim, Z. Wang, E. Shechtman, and T. Darrell. Multi-content GAN for few-shot font style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7564–7573, 2018.

X. Bai, L. J. Latecki, and W.-Y. Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE transactions on pattern analysis and machine intelligence*, 29(3), 2007.

E. Balashova, A. H. Bermano, V. G. Kim, S. DiVerdi, A. Hertzmann, and T. Funkhouser. Learning a stroke-based representation for fonts. *Computer Graphics Forum*, 38(1):429–442, 2019.

I. Baran, J. Lehtinen, and J. Popović. Sketching clothoid splines using shortest paths. In *Computer Graphics Forum*, volume 29, pages 655–664. Wiley Online Library, 2010.

C. Barber, D. Dobkin, and H. Huhdanpaa. The QuickHull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996. http://www.qhull.org.

B. Barsky and T. DeRose. Geometric continuity of parametric curves: Three equivalent characterizations. *IEEE Computer Graphics and Applications*, 9(6):60–69, Nov 1989.

P. Baudelaire and M. Gangnet. Planar maps: An interaction paradigm for graphic design. In *ACM SIGCHI Bulletin*, volume 20, pages 313–318. ACM, 1989.

S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

A. Belyaev and S. Yoshizawa. On evolute cusps and skeleton bifurcations. In *International Conference on Shape Modeling and Applications*, pages 134–140. IEEE, 2001.

J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

D. Berio and F. F. Leymarie. Computational models for the analysis and synthesis of graffiti tag strokes. In P. Rosin, editor, *Computational Aesthetics*, pages 35–47. Eurographics Association, 2015.

D. Berio, S. Calinon, and F. F. Leymarie. Learning dynamic graffiti strokes with a compliant robot. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3981–3986. IEEE, 2016.

D. Berio, M. Akten, F. Fol Leymarie, M. Grierson, and R. Plamondon. Calligraphic stylisation learning with a physiologically plausible model of movement and recurrent neural networks. In *Proc. of 4th Int'l Conf. on Movement Computing (MOCO)*, London, UK, 2017a.

D. Berio, S. Calinon, and F. Fol Leymarie. Generating calligraphic trajectories with model predictive control. In *Proceedings of Graphics Interface*, Edmonton, Canada, May 2017b. Canadian Human-Computer Communications Society.

D. Berio, S. Calinon, and F. F. Leymarie. Dynamic graffiti stylisation with stochastic optimal control. In *Proceedings of the 4th International Conference on Movement Computing*. Association for Computing Machinery, 2017c. Article no. 18.

D. Berio, F. Fol Leymarie, and R. Plamondon. Computer aided design of handwriting trajectories with the kinematic theory of rapid human movements. In *18th Biennial Conference of the International Graphonomics Society*, 2017d.

D. Berio, F. F. Leymarie, and R. Plamondon. Expressive curve editing with the sigma lognormal model. In *Proceedings of the 39th Annual European Association for Computer Graphics Conference: Short Papers*, pages 33–36. Eurographics Association, 2018a.

D. Berio, F. F. Leymarie, and R. Plamondon. Kinematic reconstruction of calligraphic traces from shape features. In *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, volume 1, pages 762–767, 2018b.

D. Berio, P. Asente, J. Echevarria, and F. Fol Leymarie. Sketching and layering graffiti primitives. In *8th ACM/Eurographics Expressive Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pages 51–59, 2019.

D. Berio, F. Fol Leymarie, and S. Calinon. Interactive generation of calligraphic trajectories from Gaussian mixtures. In N. Bouguila and W. Fan, editors, *Mixture Models and Applications*, Unsupervised and Semi-Supervised Learning Series, chapter 2, pages 23–38. Springer, 2020a. doi: 10.1007/978-3-030-23876-6_2.

D. Berio, F. F. Leymarie, and R. Plamondon. Kinematics reconstruction of static calligraphic traces from curvilinear shape features. In *The Lognormality Principle and its Applications in e-Security, e-Learning and e-Health*, chapter 11, pages 237–268. World Scientific, Nov. 2020b.

N. A. Bernstein. *The co-ordination and regulation of movements*. Pergamon Press Ltd., 1967.

E. Bertolazzi and M. Frego. Fast and accurate G^1 fitting of clothoid curves. *arXiv preprint arXiv:1305.6644*, 2013.

H. Bezine, A. M. Alimi, and N. Sherkat. Generation and analysis of handwriting script with the beta-elliptic model. *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, 8(2):515–520, 2004.

I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115—147, Apr 1987.

T. C. Biedl, C. Grimm, L. Palios, J. R. Shewchuk, and S. Verdonschot. Realizing farthest-point Voronoi diagrams. In *Proceedings of the 28th Canadian Conference on Computational Geometry (CCCG)*, pages 48–56, 2016.

C. M. Bishop. Mixture density networks. Technical report, Aston University, 1994.

E. Bizzi and A. Polit. Processes controlling visually evoked movements. *Neuropsychologia*, 17(2):203–213, 1979.

E. Bizzi, F. A. Mussa-Ivaldi, and S. Giszter. Computations underlying the execution of movement: A biological perspective. *Science*, 253(5017):287–291, 1991.

E. Bizzi, N. Hogan, F. A. Mussa-Ivaldi, and S. Giszter. Does the nervous system use equilibrium-point control to guide single and multiple joint movements? *Behavioral and brain sciences*, 15(04):603–613, 1992.

A. Blanchard. L'hypothèse de l'unité de Ductus en paléographie papyrologique. *Scrittura e civiltà*, (23): 5–27, 1999.

H. Blum. An associative machine for dealing with the visual field and some of its biological implications. In *Biological prototypes and synthetic systems*, pages 244–260. Springer, 1962.

H. Blum. A Transformation for Extracting New Descriptors of Shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967. Proceedings of a symposium held in 1964.

H. Blum. Biological shape and visual science (part I). *Journal of Theoretical Biology*, 38(2):205–287, 1973.

H. Blum and R. N. Nagel. Shape description using weighted symmetric axis features, 1978. ISSN 00313203.

M. Boden. *The creative mind: Myths and mechanisms*. Routledge, 2003.

M. Brady and H. Asada. Smoothed local symmetries and their implementation. *The International Journal of Robotics Research*, 3(3):36–61, 1984.

J.-J. Brault and R. Plamondon. Segmenting handwritten signatures at their perceptually important points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):953–957, September 1993a.

J.-J. Brault and R. Plamondon. A complexity measure of handwritten curves: Modeling of dynamic signature forgery. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):400–413, March/April 1993b.

S. E. Brennan. Caricature generator: The dynamic exaggeration of faces by computer. *Leonardo*, 18(3): 170–178, 1985.

G. S. Briem, K. Ackoff, A. Blackman, T. Botts, G. S. Briem, E. Clayton, R. Cusick, S. Day, M. Drogin, J. Evans, et al. Special issue on calligraphy. *Visible Language*, 17(1), 1983.

J. L. Brooks. Traditional and new principles of perceptual grouping. In J. Wagemans, editor, *The Oxford Handbook of Perceptual Organization*, pages 57–87. Oxford University Press, 2015.

A. E. Bryson. *Dynamic optimization*. Addison Wesley Longman Menlo Park, 1999.

A. Bucksch and R. Lindenbergh. CAMPINO — a skeletonization method for point cloud processing. *ISPRS journal of photogrammetry and remote sensing*, 63(1):115–127, 2008.

D. Bullock, S. Grossberg, and C. Mannes. A neural network model for cursive script production. *Biological Cybernetics*, 70(1):15–28, 1993.

R. W. Burkhardt. Lamarck, evolution, and the inheritance of acquired characters. *Genetics*, 194(4):793–805, 2013.

S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016a.

S. Calinon. Stochastic learning and control in multiple coordinate systems. In *Intl Workshop on Human-Friendly Robotics*, number EPFL-CONF-223744, 2016b.

S. Calinon. Mixture models for the analysis, edition, and synthesis of continuous time series. In N. Bouguila and W. Fan, editors, *Mixture Models and Applications*, pages 39–57. Springer, Cham, 2019.

S. Calinon and D. Lee. Learning control. In P. Vadakkepat and A. Goswami, editors, *Humanoid Robotics: a Reference*, pages 1261–1312. Springer, 2019.

N. D. Campbell and J. Kautz. Learning a manifold of fonts. *ACM Transactions on Graphics (TOG)*, 33(4), 2014. doi: 10.1145/2601097.2601212. Article no. 91.

R. Chamberlain, K. Chana, G. Orgs, D. Berio, and F. F. Leymarie. The naturalness of artistic mark-making predicts aesthetic value. In *Visual Science of Art Conference, Leuven, Belgium*, 2019.

R. Chamberlain, C. Mullin, D. Berio, F. F. Leymarie, and J. Wagemans. Aesthetics of graffiti: Comparison to text-based and pictorial artforms. *Empirical Studies of the Arts*, 2020.

F. Chazal and A. Lieutier. The "λ-medial axis". *Graphical Models*, 67(4):304–331, 2005.

B. Chazelle and H. Edelsbrunner. An improved algorithm for constructing kth-order Voronoi diagrams. *IEEE Transactions on Computers*, C-36(11):1349–1354, 1987.

H.-I. Chen, T.-J. Lin, X.-F. Jian, I. Shen, B.-Y. Chen, et al. Data-driven handwriting synthesis in a conjoined manner. *Computer Graphics Forum*, 34(7):235–244, 2015.

X. Chen, Z. Lian, Y. Tang, and J. Xiao. An automatic stroke extraction method using manifold learning. In *Proceedings of the European Association for Computer Graphics: Short Papers*, EG '17, pages 65–68. Eurographics Association, 2017.

H. Choi, S.-J. Cho, and J. H. Kim. Generation of handwritten characters with Bayesian network based on-line handwriting recognizers. In *null*, page 995. IEEE, 2003.

H. Choi, S. J. Cho, and J. H. Kim. Writer dependent online handwriting generation with Bayesian network. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 130–135. IEEE, 2004.

S. Collins, A. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

J. L. Coolidge. The unsatisfactory story of curvature. *The American Mathematical Monthly*, 59(6):375–379, 1952.

M. Cooper and H. Chalfant. *Subway Art*. Holt, Rinehart and Winston, 1984.

C. H. Cox, P. Coueignoux, B. Blesser, and M. Eden. Skeletons: A link between theoretical and physical letter descriptions. *Pattern Recognition*, 15(1):11–22, 1982.

R. P. C. D. A. Craveiro. The influence of graffiti writing in contemporary typography. *SAUC — Street Art and Urban Creativity Scientific Journal*, 3(2):65–83, 2017.

L. Crnkovic-Friis and L. Crnkovic-Friis. Generative choreography using deep learning. In F. Pachet, A. Cardoso, V. Corruble, and F. Ghedini, editors, *Proceedings of the Seventh International Conference on Computational Creativity (ICCC)*, pages 271–277, 2016.

G. R. Cross and A. K. Jain. Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–39, 1983.

C. Curtis. Graffiti archeology. http://grafarc.org, 2002.

C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 421–430, 1997.

B. Dalstein, R. Ronfard, and M. Van de Panne. Vector graphics complexes. *ACM Transactions on Graphics (TOG)*, 33(4):133, 2014.

A. d'Avella, P. Saltiel, and E. Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature neuroscience*, 6(3):300–308, 2003.

E. Dayan, A. Casile, N. Levit-Binnun, M. A. Giese, T. Hendler, and T. Flash. Neural representations of kinematic laws of motion: Evidence for action-perception coupling. *Proceedings of the National Academy of Sciences*, 104(51):20582–20587, 2007.

M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

C. de Boor. A practical guide to splines. *Applied Mathematical Sciences*, 1978. doi: 10.1007/978-1-4612-6333-3.

L. H. de Figueiredo. Adaptive sampling of parametric curves. In A. W. Paeth, editor, *Graphics Gems V*, chapter IV.4, pages 173–178. Academic Press, 1995.

H. De Preester. *Moving Imagination: Explorations of gesture and inner movement*, volume 89. John Benjamins Publishing, 2013.

C. De Stefano, C. D'Elia, M. Garruto, A. Marcelli, and A. S. Di Freca. A wavelet based curve decomposition method for on-line handwriting. *Advances in Graphonomics: Proceedings of IGS*, 2005.

J. De Winter and J. Wagemans. Contour-based object identification and segmentation: Stimuli, norms and data, and software tools. *Behavior Research Methods, Instruments, & Computers*, 36(4):604–624, 2004.

J. De Winter and J. Wagemans. Segmentation of object outlines into parts: A large-scale integrative study. *Cognition*, 99(3):275–325, 2006.

J. De Winter and J. Wagemans. The awakening of Attneave's sleeping cat: Identification of everyday objects on the basis of straight-line versions of outlines. *Perception*, 37(2):245–270, 2008a.

J. De Winter and J. Wagemans. Perceptual saliency of points along the contour of everyday objects: A large-scale study. *Perception and Psychophysics*, 70(1):50–64, 2008b.

R. A. DeCarlo. *Linear systems: A state variable approach with numerical implementation*. Prentice-Hall, Inc., 1989.

D. Del Vecchio, R. M. Murray, and P. Perona. Decomposition of human motion into dynamics-based primitives with application to drawing tasks. *Automatica*, 39(12):2085–2098, 2003.

J. Denier and J. P. Thuring. The guiding of human writing movements. *Biological Cybernetics*, 2(4):145–148, 1965.

O. Deussen, T. Lindemeier, S. Pirk, and M. Tautzenberger. Feedback-guided stroke placement for a painting machine. In *8th Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 25–33, 2012.

S. Deutsch and G. Medioni. Learning the geometric structure of manifolds with singularities using the tensor voting graph. *Journal of Mathematical Imaging and Vision*, 57(3):402–422, 2017.

M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer, 2013. Updated and revised second edition.

M. Diaz-Cabrera, A. Fischer, M. A. Ferrer, and R. Plamondon. Dynamic signature verification system based on one real signature. *IEEE Transactions on Cybernetics*, 48(1):228–239, 2018.

P. Dierckx. An algorithm for smoothing, differentiation and integration of experimental data using spline functions. *Journal of Computational and Applied Mathematics*, 1(3):165–184, 1975.

F. Dietrich. Visual intelligence: The first decade of computer art (1965-1975). *Leonardo*, 19(2):159–169, 1986.

A. R. Dill, M. D. Levine, and P. B. Noble. Multiple resolution skeletons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):495–504, July 1987.

J. B. Dingwell, C. D. Mah, and F. A. Mussa-Ivaldi. Experimentally confirmed mathematical model for human control of a non-rigid object. *Journal of Neurophysiology*, 91(3):1158–1170, 2004.

S. DiVerdi. A brush stroke synthesis toolbox. In *Image and video-based artistic stylisation*, pages 23–44. Springer, 2013.

M. Djioua and R. Plamondon. An interactive system for the automatic generation of huge handwriting databases from a few specimens. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008a.

M. Djioua and R. Plamondon. A new methodology to improve myoelectric signal processing using handwriting. In *International Conference on Frontiers in Handwriting Recognition, Montreal*, pages 112–117, 2008b.

M. Djioua and R. Plamondon. Studying the variability of handwriting patterns using the kinematic theory. *Human movement science*, 28(5):588–601, 2009.

M. Djioua and R. Plamondon. The limit profile of a rapid movement velocity. *Human Movement Science*, 29(1):48 – 61, 2010.

E. H. Dooijes. Analysis of handwriting movements. *Acta Psychologica*, 54(1):99–114, 1983.

B. Dresp-Langley. 2D geometry predicts perceived visual curvature in context-free viewing. *Computational Intelligence and Neuroscience*, 2015.

B. Durix, G. Morin, S. Chambon, J.-L. Mari, and K. Leonard. One-step compact skeletonization. In *Eurographics (Short Papers)*, pages 21–24, 2019.

T. Dwyer, N. Hurst, and D. Merrick. A fast and simple heuristic for metro map path simplification. In *International Symposium on Visual Computing*, pages 22–30. Springer, 2008.

S. Edelman and T. Flash. A model of handwriting. *Biological cybernetics*, 57(1-2):25–36, 1987.

M. Egerstedt and C. Martin. *Control Theoretic Splines: Optimal Control, Statistics, and Path Planning*. Princeton University Press, 2009.

M. B. Egerstedt, C. F. Martin, et al. A note on the connection between Bezier curves and linear optimal control. *IEEE transactions on automatic control*, 49(10):1728–1731, 2004.

J. H. Elder. Bridging the dimensional gap: Perceptual organization of contour into two-dimensional shape. In J. Wagemans, editor, *The Oxford Handbook of Perceptual Organization*, pages 207–235. Oxford University Press, 2015.

S. E. Engelbrecht. Minimum principles in motor control. *Journal of Mathematical Psychology*, 45(3): 497–542, 2001.

U. A. Ernst, S. Mandon, N. Schinkel-Bielefeld, S. D. Neitzel, A. K. Kreiter, and K. R. Pawelzik. Optimality of human contour integration. *PLOS Computational Biology*, 8(5):1–17, 2012.

G. Farin. The Bernstein form of a Bézier curve. In G. Farin, editor, *Curves and Surfaces for CAGD (Fifth Edition)*, The Morgan Kaufmann Series in Computer Graphics, pages 57 – 79. Morgan Kaufmann, fifth edition edition, 2002.

G. Farin, G. Rein, N. Sapidis, and A. Worsey. Fairing cubic B-spline curves. *Computer Aided Geometric Design*, 4(1–2):91—103, Jul 1987.

J.-D. Favreau, F. Lafarge, and A. Bousseau. Fidelity vs. simplicity: A global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG)*, 35(4), 2016. Article no. 120.

A. Feldman. Functional tuning of the nervous system with control of movement of maintenance of a steady posture of movement or maintenance of a steady posture. II. Controllable parameters of the muscles. *Biofizika*, 11:498–508, 1966.

J. Feldman and M. Singh. Information along contours and object boundaries. *Psychological Review*, 112(1):243–252, 2005. doi: 10.1037/0033-295X.112.1.243.

J. Feldman and M. Singh. Bayesian estimation of the shape skeleton. *Proceedings of the National Academy of Sciences*, 103(47):18014–18019, 2006.

M. Ferrer, M. Diaz-Cabrera, A. Morales, et al. Static signature synthesis: A neuromotor inspired approach for biometrics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(3):667–680, 2015.

M. A. Ferrer, M. Diaz-Cabrera, and A. Morales. Static signature synthesis: A neuromotor inspired approach for biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):667–680, 2015.

M. A. Ferrer, M. Diaz, C. Carmona-Duarte, and R. Plamondon. iDeLog: Iterative dual spatial and kinematic extraction of sigma-lognormal parameters. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):114–125, 2018.

A. Ferri. *Teoria del writing, La ricerca dello stile.* Professional Dreamers, 2016.

D. J. Field, A. Hayes, and R. F. Hess. Contour integration by the human visual system: Evidence for a local "association field". *Vision Research*, 33(2):173–193, Jan 1993. ISSN 0042-6989.

C. Firestone and B. J. Scholl. "please tap the shape, anywhere you like" shape skeletons in human vision revealed by an exceedingly simple measure. *Psychological science*, 25(2):377–386, 2014.

A. Fischer, R. Plamondon, C. O'Reilly, and Y. Savaria. Neuromuscular representation and synthetic generation of handwritten whiteboard notes. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 222–227. IEEE, 2014.

P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381, 1954.

P. M. Fitts and M. I. Posner. *Human performance.* Brooks/Cole, 1967.

T. Flash. *Organizing principles underlying the formation of arm trajectories.* PhD thesis, Massachusetts Institute of Technology, 1983.

T. Flash and A. A. Handzel. Affine differential geometry analysis of human arm movements. *Biological cybernetics*, 96(6):577–601, 2007.

T. Flash and E. Henis. Arm trajectory modifications during reaching towards visual targets. *Journal of cognitive Neuroscience*, 3(3):220–230, 1991.

T. Flash and B. Hochner. Motor primitives in vertebrates and invertebrates. *Current opinion in neurobiology*, 15(6):660–6, 2005.

T. Flash and N. Hogan. The coordination of arm movements. *Journal of Neuroscience*, 5(7):1688–1703, 1985.

M. S. Floater and T. Surazhsky. Parameterization for curve interpolation. In *Studies in Computational Mathematics*, volume 12, pages 39–54. Elsevier, 2006.

J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics (2nd Ed. in C): Principles and Practice.* Addison-Wesley Longman Publishing Co., Inc., USA, 1995. ISBN 0201848406.

T. A. Foley and G. M. Nielson. Knot selection for parametric spline interpolation. In *Mathematical methods in computer aided geometric design*, pages 261–CP4. Elsevier, 1989.

W. C. Fong. Why Chinese painting is history. *The Art Bulletin*, 85(2):258–280, 2003.

K. Franke and S. Rose. Ink-deposition model: The relation of writing and ink deposition processes. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 173–178. IEEE, 2004.

D. Freedberg and V. Gallese. Motion, emotion and empathy in esthetic experience. *Trends in cognitive sciences*, 11(5):197–203, 2007.

F. N. Freeman. Experimental analysis of the writing movement. *Psychological Monographs: General and Applied*, 17(4):1–57, 1914.

W. T. Freeman, J. B. Tenenbaum, and E. C. Pasztor. Learning style translation for the lines of a drawing. *ACM Transactions on Graphics (TOG)*, 22(1):33–46, 2003.

J. J. Freyd. Representing the dynamics of a static form. *Memory & cognition*, 11(4):342–346, 1983. ISSN 0090-502X.

J. J. Freyd. Dynamic mental representations. *Psychological review*, 94(4):427–438, 1987. doi: 10.1037/0033-295X.94.4.427.

V. Froyen, J. Feldman, and M. Singh. Bayesian hierarchical grouping: Perceptual grouping as mixture estimation. *Psychological Review*, 122(4):575–597, 2015. doi: 10.1037/a0039540.

H. Fu, S. Zhou, L. Liu, and N. J. Mitra. Animated construction of line drawings. In *ACM Transactions on Graphics (TOG)*, volume 30, pages 1–10, 2011. doi: 10.1145/2070781.2024167.

H. Fujioka and S. Miyata. Reshaping and reconstructing handwritten character typeface using dynamic font model. In *2011 Third International Conference on Intelligent Networking and Collaborative Systems*, pages 563–568, Nov 2011.

H. Fujioka, H. Kano, H. Nakata, and H. Shinoda. Constructing and reconstructing characters, words, and sentences by synthesizing writing motions. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(4):661–670, 2006.

J. Galbally, R. Plamondon, J. Fierrez, and J. Ortega-Garcia. Synthetic on-line signature generation. Part I: Methodology and algorithms. *Pattern Recognition*, 45(7):2610–2621, 2012.

V. Gallese, L. Fadiga, L. Fogassi, and G. Rizzolatti. Action recognition in the premotor cortex. *Brain*, 119 (2):593–610, 1996.

A. Galton and R. C. Meathrel. Qualitative outline theory. In *IJCAI*, pages 1061–1066, 1999.

F. Gao, G. Wei, S. Xin, S. Gao, and Y. Zhou. 2D skeleton extraction based on heat equation. *Computers & Graphics*, 74:99–108, 2018.

Y. Gao, Y. Guo, Z. Lian, Y. Tang, and J. Xiao. Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics*, 38(6):1–12, Nov 2019.

P. Garrigan and P. J. Kellman. The role of constant curvature in 2-D contour shape representations. *Perception*, 40(11):1290–1308, 2011.

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

F. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

C. Ghez, M. Favilla, M. Ghilardi, J. Gordon, R. Bermejo, and S. Pullman. Discrete and continuous planning of hand movements and isometric force trajectories. *Experimental Brain Research*, 115(2):217–233, 1997.

P. K. Ghosh and C. A. Bigelow. A formal approach to lettershape description for type design. Technical report, Department of Computer Science, Stanford University, 1983.

É. Ghys, S. Tabachnikov, and V. Timorin. Osculating curves: Around the Tait-Kneser theorem. *The Mathematical Intelligencer*, 35(1):61–66, 2013.

P. Giblin. Symmetry sets and medial axes in two and three dimensions. In *The Mathematics of Surfaces IX*, pages 306–321. Springer, 2000.

P. J. Giblin and B. B. Kimia. On the local form and transitions of symmetry sets, medial axes, and shocks. *International Journal of Computer Vision*, 54(1):143–157, 2003.

Y. Gingold, D. Salesin, and D. Zorin. Stroke-by-stroke glyph animation. Technical report, Creativity and Graphics Lab (CraGL) at George Mason University, Fairfax, Virginia, USA, 2008. https://cragl.cs.gmu.edu/fontanim/.

O. Gold and M. Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *ACM Trans. Algorithms*, 14(4), 2018. doi: 10.1145/3230734.

A. Goldberg, X. Zhu, A. Singh, Z. Xu, and R. Nowak. Multi-manifold semi-supervised learning. In *12th International Conference on Artificial Intelligence and Statistics*, pages 169–176, 2009.

E. H. Gombrich. *Art and illusion: A study in the psychology of pictorial representation*, volume 5. Phaidon London, 1977.

M. Gomez-Barrero, J. Galbally, J. Fierrez, J. Ortega-Garcia, and R. Plamondon. Enhanced on-line signature verification based on skilled forgery detection using sigma-lognormal features. In *2015 international conference on biometrics (ICB)*, pages 501–506. IEEE, 2015.

A. A. Gooch, J. Long, L. Ji, A. Estey, and B. S. Gooch. Viewing progress in non-photorealistic rendering through Heinlein's lens. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 165–171. ACM, 2010.

I. C. Graphics and Applications. Fractional invisibility. *IEEE Comput. Graph. Appl.*, 8(6):77–84, Nov. 1988.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

I. Grebert, D. G. Stork, R. Keesing, and S. Mims. Connectionist generalization for production: An example from gridfont. *Neural Networks*, 5(4):699–710, 1992.

K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A Recurrent Neural Network For Image Generation. *arXiv preprint arXiv:1502.04623*, 2015.

S. Grossberg and R. W. Paine. A neural model of cortico-cerebellar interactions during attentive imitation and predictive learning of sequential handwriting movements. *Neural Networks*, 13(8):999–1046, 2000.

D. Ha and D. Eck. A neural representation of sketch drawings. In *Sixth International Conference on Learning Representations (ICLR)*, 2018. https://arxiv.org/abs/1704.03477.

P. Haeberli. Dynadraw: A dynamic drawing technique. http://www.graficaobscura.com/dyna/, 1989.

T. S. Haines, O. M. Aodha, and G. J. Brostow. My text in your handwriting. *ACM Transactions on Graphics (TOG)*, 35(3), 2016. Article no. 26.

P. C. Hansen. The L-curve and its use in the numerical treatment of inverse problems. In *Computational Inverse Problems in Electrocardiology*, pages 119–142. WIT Press, 2000.

C. M. Harris and D. M. Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394(6695):780–784, 1998.

F. Haugen. Discrete-time signals and systems. 2005. URL `http://techteach.no/publications/discretetime_signals_systems/discrete.pdf`.

S. Havemann, J. Edelsbrunner, P. Wagner, and D. Fellner. Curvature-controlled curve editing using piecewise clothoid curves. *Computers & Graphics*, 37(6):764–773, 2013.

H. Hayashi, K. Abe, and S. Uchida. GlyphGAN: Style-consistent font generation based on generative adversarial networks. *Knowledge-Based Systems*, 186, 2019.

P. J. Hayes and M. Leyton. Processes at discontinuities. In *IJCAI*, pages 1267–1272, 1989.

M. A. Heald. Rational approximations for the Fresnel integrals. *Mathematics of Computation*, 44(170):459–459, 1985. ISSN 0025-5718.

K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. In *22nd International Conference on Machine learning (ICML)*, pages 297–304. ACM, 2005.

M. Hendrickx and J. Wagemans. A critique of Leyton's theory of perception and cognition. Review of Symmetry, Causality, Mind, by Michael Leyton, 1999.

F. M. Henry and D. E. Rogers. Increased response latency for complicated movements and a memory drum theory of neuromotor reaction. *Research Quarterly. American Association for Health, Physical Education and Recreation*, 31(3):448–458, 1960.

A. Hertzmann. Non-photorealistic rendering and the science of art. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 147–157. ACM, 2010.

A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, page 327–340, 2001.

A. Hertzmann, N. Oliver, B. Curless, and S. M. Seitz. Curve analogies. In *Rendering Techniques*, pages 233–246, 2002.

J. Herz, R. D. Hersch, and J. Gonczarowski. Coherent processing of character skeletal forms. *Computers and Graphics*, 21(6):727–736, 1997.

J. Hespanha. Lecture notes on lqr/lqg controller design. staff.uz.zgora.pl/wpaszke/materialy/kss/lqrnotes.pdf, 2005.

J. D. Hobby. Smooth, easy to compute interpolating splines. *Discrete and Computational Geometry*, 1 (2):123—140, Jun 1986.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

B. Hoff. A model of duration in normal and perturbed reaching movement. *Biological Cybernetics*, 71 (6):481–488, 1994.

D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18(1-3):65–96, 1984. doi: 10.1016/0010-0277(84)90022-2.

D. D. Hoffman and M. Singh. Salience of visual parts. *Cognition*, 63(1):29–78, 1997. doi: 10.1016/S0010-0277(96)00791-3.

D. Hofstadter. Metamagical themas: Variations on a theme as the essence of imagination. *Scientific American*, 247(4):14–21, 1982.

D. Hofstadter. *Metamagical themas: Questing for the essence of mind and pattern*. Basic Books, New York, 1985. ISBN 978-0465045662.

D. Hofstadter, G. McGraw, et al. Letter spirit: An emergent model of the perception and creation of alphabetic style. In *Technical Report 68, Center for Research on Concepts and Cognition*, 1993.

N. Hogan. Control and coordination of voluntary arm movements. In *American Control Conference, 1982*, pages 522–528. IEEE, 1982.

A. Holland Michael. KATSU shows you how to make a graffiti drone. `https://www.vice.com/en_us/article/mvxedv/katsu-shows-you-how-to-make-a-graffiti-drone-456`, 2015.

J. M. Hollerbach. An oscillation theory of handwriting. *Biological Cybernetics*, 39(2):139–156, 1981.

D. H. House and M. Singh. Line drawing as a dynamic process. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 351–360, 2007.

S. C. Hsu and I. H. H. Lee. Drawing and animation using skeletal strokes. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 109–118, New York, New York, USA, 1994.

C. Hu and R. D. Hersch. Parameterizable fonts based on shape components. *IEEE Computer Graphics and Applications*, 21(3):70–85, 2001. doi: 10.1109/38.920629.

J. Hulleman, W. Te Winkel, and F. Boselie. Concavities as basic features in visual search: Evidence from search asymmetries. *Perception & Psychophysics*, 62(1):162–174, 2000.

T. Igarashi and J. Mitani. Apparent layer operations for the manipulation of deformable objects. In *ACM Transactions on Graphics (TOG)*, volume 29, page 110, 2010.

T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification. *ACM SIGGRAPH 2007 courses on - SIGGRAPH '07*, 2007. doi: 10.1145/1281500.1281529.

W. R. Jack. On the analysis of voluntary muscular movements by certain new instruments. *Proceedings of the Royal Society of London*, 57(340-346):477–481, 1894.

E. J. Jakubiak, R. N. Perry, and S. F. Frisken. An improved representation for stroke-based fonts. In *ACM SIGGRAPH 2006 Sketches*, 2006.

K. H. James and I. Gauthier. Letter processing automatically recruits a sensory-motor brain network. *Neuropsychologia*, 44(14):2937–2949, 2006.

M. Jeannerod. Mental imagery in the motor context. *Neuropsychologia*, 33(11):1419–1432, 1995.

M. I. Jordan and D. M. Wolpert. Computational motor control, 1999.

G. Kanizsa. *Organization in Vision: Essays on Gestalt Perception*. Praeger, 1979.

H. Kano, H. Fujioka, and K. Inoue. Discrete-time control systems approach for optimal smoothing splines. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 356–361, 2005. doi: 10.1109/CDC.2005.1582181.

H. S. Kao, R. Hoosain, and G. Van Galen. *Graphonomics: Contemporary research in handwriting*. Elsevier, 1986.

P. Karow. *Digital Typefaces: Description and Formats*. Springer, 1994.

R. Kaushik. Cy Twombly (gesture, space, and writing). In H. De Preester, editor, *Moving imagination, explorations of gesture and inner movement in the arts*, pages 235—246. John Benjamins Publishing Company, 2013.

S. W. Keele and J. J. Summers. The structure of motor programs. *Motor control: Issues and trends*, pages 109–142, 1976.

P. J. Kellman and P. Garrigan. Segmentation, grouping, and shape: Some hochbergian questions. 2007.

P. J. Kellman and T. F. Shipley. A theory of visual interpolation in object perception. *Cognitive Psychology*, 23(2):141—221, Apr 1991.

J. Kelso and E. Saltzman. Motor control: Which themes do we orchestrate? *Behavioral and Brain Sciences*, 5(04):554–557, 1982.

M. G. Kendall and J. K. Ord. *Time Series*. 06 1993. ISBN 9780340593271.

J. C. Keough. *Graffiti Research Lab: Bridging the Canonical and the Criminal*. PhD thesis, The Graduate School, Stony Brook University: Stony Brook, NY., 2010.

B. Kim, O. Wang, A. C. Öztireli, and M. Gross. Semantic segmentation for line drawing vectorization using neural networks. *Computer Graphics Forum*, 37(2):329–338, 2018. doi: 10.1111/cgf.13365.

B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum*, 38(2):59–70, 2019. doi: 10.1111/cgf.13619.

B. Kimia, I. Frankel, and A.-M. Popescu. Euler spiral for shape completion. *International journal of computer vision*, 54:159–182, 2003.

B. B. Kimia. On the role of medial geometry in human vision. *Journal of Physiology–Paris*, 97(2-3): 155–190, 2003.

B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker. Shapes, shocks, and deformations I: The components of two-dimensional shape and the reaction-diffusion space. *International journal of computer vision*, 15(3):189–224, 1995.

R. Kimmel, D. Shaked, N. Kiryati, and A. M. Bruckstein. Skeletonization via distance maps and level sets. *Computer Vision and Image Understanding*, 62(3):382 – 391, 1995.

J. Kimvall. Bad graffiti art gone good street art. 2007. URL `http://www.academia.edu/1121025/Bad_Graffiti_Art_Gone_Good_Street_Art`.

J. Kimvall. *The G-word*. Dokument, Stockholm, 2014. ISBN 9789185639687.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, pages 1–13, 2014.

S. Kishore. Make me a hanzi dataset. `https://github.com/skishore/makemeahanzi`, 2018. URL `www.skishore.me/makemeahanzi/`.

P. Klee. *Paul Klee: The thinking eye. The notebooks of Paul Klee*, volume 15. G. Wittenborn, 1961.

D. E. Knuth. Mathematical typography. *Bulletin of the American Mathematical Society*, 1(2):337–373, 1979. doi: 10.1090/S0273-0979-1979-14598-1.

D. E. Knuth. Mathematical typography. In *Digital Typography*, pages 19–65. Csli Publications, 1999.

J. Koenderink and A. van Doorn. The Structure of Visual Spaces. *Journal of Mathematical Imaging and Vision*, 31(2-3):171–187, 2008.

J. J. Koenderink. The structure of images. *Biological cybernetics*, 50(5):363–370, 1984.

J. J. Koenderink. *Solid shape*. MIT press, 1990.

J. J. Koenderink. Geometry of imaginary spaces. *Journal of Physiology-Paris*, 106(5):173–182, 2012.

I. Kovács, Á. Fehér, and B. Julesz. Medial-point description of shape: A representation for action coding and its psychophysical correlates. *Vision research*, 38(15):2323–2333, 1998.

R. T. Krampe, R. Engbert, and R. Kliegl. Representational models and nonlinear dynamics: Irreconcilable approaches to human movement timing and coordination or two sides of the same coin? introduction to the special issue on movement timing and coordination. *Brain and Cognition*, 48(1): 1–6, 2002.

A. Kuijper, O. F. Olsen, P. Giblin, and M. Nielsen. Alternative 2D shape representations using the symmetry set. *Journal of Mathematical Imaging and Vision*, 26:127–147, 2006.

J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):866–885, 2013.

F. Lacquaniti, C. Terzuolo, and P. Viviani. The law relating the kinematic and figural aspects of drawing movements. *Acta psychologica*, 54(1):115–130, 1983.

B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, pages 2526–2534, 2013.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

B. Lamiroy, T. Bouville, J. Blégean, H. Cao, S. Ghamizi, R. Houpin, and M. Lloyd. Re-typograph phase I: A proof-of-concept for typeface parameter extraction from historical documents. In E. K. Ringger and B. Lamiroy, editors, *Document Recognition and Retrieval XXII*, volume 9402, pages 80–91. International Society for Optics and Photonics, SPIE, 2015.

K. Lang and M. Alexa. The Markov pen: Online synthesis of free-hand drawing styles. In *Proceedings of the workshop on Non-Photorealistic Animation and Rendering*, pages 203–215. Eurographics Association, 2015.

K. S. Lashley. *The problem of serial order in behavior*. Bobbs-Merrill, 1951.

L. J. Latecki and R. Lakämper. Discrete approach to curve evolution. In *Mustererkennung 1998*, pages 85–92. Springer, 1998.

R. Leavitt. *Artist and Computer*. Harmony Books, 1976.

H. Leder, S. Bär, and S. Topolinski. Covert painting simulations influence aesthetic appreciation of artworks. *Psychological Science*, 23(12):1479–1481, 2012.

D.-H. Lee and H.-G. Cho. The beta-velocity model for simulating handwritten korean scripts. In *Electronic Publishing, Artistic Imaging, and Digital Typography*, pages 252–264. Springer, 1998.

E. T. Lee. Choosing nodes in parametric curve interpolation. *Computer-Aided Design*, 21(6):363–370, 1989.

J. Lehni. Hektor. `http://hektor.ch/`, 2004.

L. A. Leiva, D. Martín-Albo, and R. Plamondon. Gestures à go go: Authoring synthetic human-like stroke gestures using the kinematic theory of rapid movements. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(2):15, 2016.

L. A. Leiva, D. Martín-Albo, and R. Plamondon. The kinematic theory produces human-like stroke gestures. *Interacting with Computers*, 29(4):552–565, July 2017.

K. Leonard, G. Morin, S. Hahmann, and A. Carlier. A 2D shape structure for decomposition and part similarity. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3216–3221. IEEE, 2016.

R. Levien. The Euler spiral: A mathematical history. *Opera*, pages 1–14, 2008.

R. Levien. *From Spiral to Spline: Optimal Techniques in Interactive Curve Design*. PhD thesis, EECS Department, University of California, Berkeley, December 2009a. PhD thesis, EECS Department, University of California, Berkeley.

R. Levien and C. H. Séquin. Interpolating splines: Which is the fairest of them all? *Computer-Aided Design and Applications*, 6(1):91–102, 2009.

R. L. Levien. From spiral to spline: Optimal techniques in interactive curve design. 2009b.

G. Levin, J. Feinberg, and C. Curtis. The alphabet synthesis machine. Technical report, 2013. URL `http://life.flong.com/storage/pdf/reports/alphabet_report.pdf`.

F. Leymarie and M. D. Levine. Curvature morphology. Technical report, McGill University, Montreal, Canada, 1988.

F. Leymarie and M. D. Levine. Shape features using curvature morphology. In D. P. Casasent, editor, *Proc. of the SPIE Conf. on Intelligent Robots and Computer Vision VIII: Algorithms and Techniques*, volume SPIE–1192, part 2, pages 536–547, Philadelphia, PA, U.S.A., Nov. 1989. SPIE.

F. Leymarie and M. D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):56–75, 1992.

F. F. Leymarie. Thoughts on shape. In L. Albertazzi, editor, *Visual Thought*, volume 67 of *Advances in Consciousness Research*, pages 303–350. John Benjamins Publishing Company, 2006.

F. F. Leymarie and P. Aparajeya. Medialness and the perception of visual art. *Art & Perception*, 5(2): 169–232, 2017.

F. F. Leymarie and B. B. Kimia. The shock scaffold for representing 3D shape. In *International workshop on visual form*, pages 216–227. Springer, 2001.

F. F. Leymarie and B. B. Kimia. The medial scaffold of 3D unorganized point clouds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):313–330, 2007.

M. Leyton. Symmetry-curvature duality. *Computer Vision, Graphics, and Image Processing*, 38(3):327–341, 1987.

M. Leyton. A process-grammar for shape. *Artificial Intelligence*, 34(2):213–247, March 1988.

M. Leyton. Inferring causal history from shape. *Cognitive Science*, 13(3):357–387, Sep 1989.

M. Leyton. Group theory and architecture. *Nexus Network Journal*, 3(2):39—58, Sep 2001a. ISSN 1522-4600.

M. Leyton. *A generative theory of shape*, volume 2145. Springer, 2001b.

M. Leyton. *The structure of paintings*. Springer, 2006.

M. Leyton. *Process grammar: The basis of morphology*. Springer Science & Business Media, 2012.

H. Li, H. Zhang, Y. Wang, J. Cao, A. Shamir, and D. Cohen-Or. Curve style analysis in a set of shapes. In *Computer Graphics Forum*, volume 32, pages 77–88. Wiley Online Library, 2013.

X. Li, M. Parizeau, and R. Plamondon. Segmentation and reconstruction of on-line handwritten scripts. *Pattern recognition*, 31(6):675–684, 1998.

Z. Lian and J. Xiao. Automatic shape morphing for Chinese characters. In *SIGGRAPH Asia 2012 Technical Briefs*, 2012.

Z. Lian, B. Zhao, X. Chen, and J. Xiao. EasyFont: A style learning-based system to easily build your large-scale handwriting fonts. *ACM Transactions on Graphics (TOG)*, 38(1):1–18, 2018.

A. Lieutier. Any open bounded subset of R^n has the same homotopy type than its medial axis. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 65–75, 2003.

G. Liu, Z. Xi, and J.-M. Lien. Dual-space decomposition of 2D complex shapes. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4154–4161, Jun 2014.

D. Llorens Piñana et al. The UJIpenchars database: A pen-based database of isolated handwritten characters. In N. Calzolari et al., editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA), may 2008.

M. Longcamp, J. L. Anton, M. Roth, and J. L. Velay. Visual presentation of single letters activates a premotor area involved in writing. *NeuroImage*, 19(4):1492–1500, 2003.

M. Longcamp, T. Tanskanen, and R. Hari. The imprint of action: Motor cortex involvement in visual perception of handwritten letters. *NeuroImage*, 33(2):681–688, 2006.

M. Longcamp, Y. Hlushchuk, and R. Hari. Neural correlates of the visual perception of handwritten letters. In *Advances in Graphonomics: Proceedings of IGS 2009*, pages 194–197, 2009.

M. Longcamp, Y. Hlushchuk, and R. Hari. What differs in visual recognition of handwritten vs. printed letters? An fMRI study. *Human brain mapping*, 32(8):1250–1259, 2011.

R. G. Lopes, D. Ha, D. Eck, and J. Shlens. A learned representation for scalable vector graphics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7930–7939, 2019.

M. Ltaief, H. Bezine, and A. M. Alimi. A neuro-beta-elliptic model for handwriting generation movements. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 803–808. IEEE, 2012.

J. Lu, F. Yu, A. Finkelstein, and S. DiVerdi. HelpingHand: Example-based stroke stylization. *ACM Transactions on Graphics (TOG)*, 31(4):46, 2012.

J. Lu, C. Barnes, C. Wan, P. Asente, R. Mech, and A. Finkelstein. Decobrush: Drawing structured decorative patterns by example. *ACM Transactions on Graphics (TOG)*, 33(4):90, 2014.

D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001. doi: 10.1109/38.920624.

L. Luo, C. Shen, X. Liu, and C. Zhang. A computational model of the short-cut rule for 2D shape decomposition. *IEEE Transactions on Image Processing*, 24(1):273–283, 2015.

P. Lyu, X. Bai, C. Yao, Z. Zhu, T. Huang, and W. Liu. Auto-encoder guided GAN for Chinese calligraphy synthesis. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1095–1100, 2017.

F. J. Maarse. *The study of handwriting movement: Peripheral models and signal processing techniques*. Lisse [etc.]: Swets & Zeitlinger, 1987.

J. S. MacDonald. *Experimental studies of handwriting signals*. Citeseer, 1966.

I. S. MacKenzie and W. Buxton. Extending Fitts' law to two-dimensional tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 219–226. ACM, 1992.

D. Macrini, K. Siddiqi, and S. Dickinson. From skeletons to bone graphs: Medial abstraction for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

D. Macrini, S. Dickinson, D. Fleet, and K. Siddiqi. Bone graphs: Medial shape parsing and abstraction. *Computer Vision and Image Understanding*, 115(7):1044–1061, 2011.

E. Maggiori, H. L. Manterola, and M. del Fresno. Perceptual grouping by tensor voting: A comparative survey of recent approaches. *IET Computer Vision*, 9(2):259–277, 2015.

B. B. Mandelbrot. A case against the lognormal distribution. In *Fractals and scaling in finance*, pages 252–269. Springer, 1997.

A. Manzanera, T. Nguyena, and X. Xu. Line and circle detection using dense one-to-one Hough transforms on greyscale images. *EURASIP Journal on Image and Video Processing*, (46), December 2016.

U. Maoz, A. Berthoz, and T. Flash. Complex unconstrained three-dimensional hand movement and constant equi-affine speed. *Journal of neurophysiology*, 101(2):1002–15, 2009.

D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., USA, 1982.

A. Massad and G. Medioni. 2-D shape decomposition into overlapping parts. In *Visual Form 2001*, pages 398–409. Springer, 2001.

J. McCann and N. Pollard. Local layering. In *ACM Transactions on Graphics (TOG)*, volume 28, page 84. ACM, 2009.

J. McCormack and A. Lomas. Understanding aesthetic evaluation using deep learning. *Lecture Notes in Computer Science*, pages 118–133, 2020.

J. McCrae and K. Singh. Sketching piecewise clothoid curves. *Computers and Graphics (Pergamon)*, 33 (4):452–461, 2009.

J. McCrae and K. Singh. Neatening sketched strokes using piecewise french curves. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling - SBIM '11*, pages 141—148, 2011.

G. McGraw Jr. *Letter Spirit (part one): Emergent high-level perception of letters using fluid concepts*. PhD thesis, Indiana University, Dept. of Computer Science, Indianapolis, U.S.A., September 1995.

C. Mediavilla, A. Marshall, M. van Stone, G. Xuriguera, and D. Jackson. *Calligraphy: From calligraphy to abstract painting*. Scirpus, 1996.

Y. Meirovitch and T. Flash. Report on segmentation of trajectories into the underlying primitives and syntactic rules for their compositionality based on differential geometry approaches. Technical Report AMARSi ICT-248311, D1.4, 2013. URL http://www.amarsi-project.eu/sites/www.amarsi-project.eu/files/D14.pdf.

R. G. Meulenbroek, A. Thomassen, D. Rosebaum, L. D. Loukopoulos, and J. Vaughan. Adaptation of a reaching model to handwriting: How different effectors can produce the same written output, and other results. *Psychological Research*, 59(1):64–74, 1996.

X. Mi. Structural representation of 2D shape. Technical report, Computer Science Dept., Rutgers University, USA, 2006.

X. Mi and D. DeCarlo. Separating parts from 2D shapes using relatability. In *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007.

K. T. Miura. A general equation of aesthetic curves and its self-affinity. *Computer-Aided Design and Applications*, 3(1-4):457–464, 2006.

T. Miyazaki, T. Tsuchiya, Y. Sugaya, S. Omachi, M. Iwamura, S. Uchida, and K. Kise. Automatic generation of typographic font from small font subset. *IEEE Computer Graphics and Applications*, 40(1): 99–111, 2019.

U. Montanari. Continuous skeletons from digitized images. *Journal of the ACM*, 16(4):534–549, Oct. 1969.

P. Morasso. Spatial control of arm movements. *Experimental Brain Research*, 42(2):223–7, 1981.

P. Morasso. Understanding cursive script as a trajectory formation paradigm. *Graphonomics*, 37:137–67, 1986.

P. Morasso and F. Mussa Ivaldi. Trajectory formation and handwriting: A computational model. *Biological cybernetics*, 45(2):131–142, 1982.

P. Mordohai and G. Medioni. Dimensionality estimation, manifold learning and function approximation using tensor voting. *The Journal of Machine Learning Research*, 11:411–450, 2010.

D. Mumford. Elastica and computer vision. *Algebraic Geometry and its Applications*, pages 491—506, 1994.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

F. Mussa-Ivaldi, S. Solla, et al. Neural primitives for motion control. *Oceanic Engineering, IEEE Journal of*, 29(3):640–650, 2004.

F. A. Mussa-Ivaldi. Nonlinear force fields. In *IEEE Proc. CIRA*, pages 84–90, 1997.

F. A. Mussa-Ivaldi and E. Bizzi. Motor learning through the combination of primitives. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 355(1404):1755–69, 2000.

H. Nagasaki. Asymmetric velocity and acceleration profiles of human arm movements. *Experimental Brain Research*, 74(2):319–26, 1989.

V. Nair and G. E. Hinton. Inferring motor programs from images of handwritten digits. In *Advances in neural information processing systems*, pages 515–22, 2005.

F. Nake. Computer art: A personal recollection. In *Proceedings of the 5th conference on Creativity & cognition*, pages 54–62. ACM, 2005.

K. M. Newell and D. E. Vaillancourt. Dimensional change in motor learning. *Human movement science*, 20(4):695–715, 2001.

A. M. Noll. Human or machine: A subjective comparison of Piet Mondrian's" composition with lines"(1917) and a computer-generated picture. *The psychological record*, 1966.

M. Nöllenburg. A survey on automated metro map layout methods. In *1st Schematic Mapping Workshop*, University of Essex, UK, 2014. https://sites.google.com/site/schematicmapping/home.

G. Noordzij. *The Stroke — theory of writing*. Hyphen Press, 2005. Translated from the Dutch original of 1985 by Peter Enneson.

R. L. Ogniewicz. *Discrete Voronoi skeletons*. PhD thesis, 1992.

R. L. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 63–69, 1992.

L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

B. O'Neill. *Elementary Differential Geometry*. Academic press, 2006. Revised second edition of the 1966 original.

C. O'Reilly and R. Plamondon. Automatic extraction of sigma-lognormal parameters on signatures. In *Proc. of 11th International Conference on Frontier in Handwriting Recognition (ICFHR)*, 2008.

C. O'Reilly and R. Plamondon. Development of a sigma-lognormal representation for on-line signatures. *Pattern recognition*, 42(12):3324–3337, 2009.

J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 2 edition, 1998.

E. Oztop, M. Kawato, and M. A. Arbib. Mirror neurons: Functions, mechanisms and models. *Neuroscience letters*, 540:43–55, 2013.

R. Paine, S. Grossberg, and A. Van Gemmert. A quantitative evaluation of the AVITEWRITE model of handwriting learning. *Human movement science*, 23(6):837–860, 2004.

N. Papanelopoulos, Y. Avrithis, and S. Kollias. Revisiting the medial axis for planar shape decomposition. *Computer Vision and Image Understanding*, 179:66–78, 2019. doi: 10.1016/j.cviu.2018.10.007.

P. Parent and S. W. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):823–839, 1989.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training Recurrent Neural Networks. In *Proc. of ICML*, volume 28, pages 1310–8, 2013.

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In *Proc. of ICFHR*, pages 285–90. IEEE, 2014.

H. Q. Phan, H. Fu, and A. B. Chan. Flexyfont: Learning transferring rules for flexible typeface synthesis. *Computer Graphics Forum*, 34(7):245–256, 2015.

A. Pignocchi. How the intentions of the draftsman shape perception of a drawing. *Consciousness and Cognition*, 19(4):887–898, 2010.

S. M. Pizer, K. Siddiqi, G. Székely, J. N. Damon, and S. W. Zucker. Multiscale medial loci and their properties. *International Journal of Computer Vision*, 55(2-3):155–179, 2003.

R. Plamondon. Looking at handwriting generation from a velocity control perspective. *Acta Psychologica*, 82(1-3):89—101, Mar 1993.

R. Plamondon. A kinematic theory of rapid human movements. Part I. Movement representation and generation. *Biological cybernetics*, 72(4):295–307, 1995.

R. Plamondon and A. M. Alimi. Speed/accuracy trade-offs in target-directed movements. *Behavioral and Brain Sciences*, 20(02):279–303, 1997.

R. Plamondon and W. Guerfali. The 2/3 power law: When and why? *Acta psychologica*, 100(1):85–96, 1998a.

R. Plamondon and W. Guerfali. The generation of handwriting with delta-lognormal synergies. *Biological Cybernetics*, 78(2):119–132, 1998b.

R. Plamondon and C. M. Privitera. A neural model for generating and learning a rapid movement sequence. *Biological cybernetics*, 74(2):117–130, 1996.

R. Plamondon and C. M. Privitera. The segmentation of cursive handwriting: An approach based on off-line recovery of the motor-temporal information. *IEEE Transactions on Image Processing*, 8(1): 80–91, 1999.

R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):63–84, 2000.

R. Plamondon, M. Djioua, and C. O'Reilly. Recent developments in the study of rapid human movements with the kinematic theory. *Traitement Du Signal*, 26:377–394, 2009.

R. Plamondon, C. O'Reilly, C. Remi, and T. Duval. The lognormal handwriter: Learning, performing and declining. *Frontiers in Psychology*, 4(945), 2013.

R. Plamondon, C. O'Reilly, J. Galbally, A. Almaksour, and É. Anquetil. Recent developments in the study of rapid human movements with the kinematic theory. *Pattern Recognition Letters*, 35:225–35, 2014.

R. Plamondon et al. Modelling velocity profiles of rapid movements: A comparative study. *Biological cybernetics*, 69(2):119–28, 1993.

R. Plamondon et al. A kinematic theory of rapid human movement. Part IV. *Biological Cybernetics*, 89 (2):126–38, 2003.

A. Polit and E. Bizzi. Processes controlling arm movements in monkeys. *Science*, 201(4362):1235–1237, 1978.

F. Polyakov, R. Drori, Y. Ben-Shaul, M. Abeles, and T. Flash. A compact representation of drawing movements with sequences of parabolic primitives. *PLoS Comput Biol*, 5(7):e1000427–e1000427, 2009.

C. Ramaiah, R. Plamondonm, and V. Govindaraju. A sigma-lognormal model for handwritten text CAPTCHA generation. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 250–255. IEEE, 2014.

J. A. Rehling. *Letter Spirit (part two): Modeling Creativity in a Visual Domain.* PhD thesis, Indiana University, Dept. of Computer Science, Indianapolis, U.S.A., July 2001.

M. Resnick, B. Myers, K. Nakakoji, B. Shneiderman, R. Pausch, T. Selker, and M. Eisenberg. Design principles for tools to support creative thinking. Technical report, Carnegie Mellon University, 2005.

W. Richards and D. D. Hoffman. Codon constraints on closed 2D shapes. *Computer Vision, Graphics, and Image Processing*, 31(3):265–281, 1985.

M. J. Richardson and T. Flash. Comparing smooth arm movements with the two-thirds power law and the related segmented-control hypothesis. *The Journal of Neuroscience*, 22(18):8201–8211, 2002.

B. Rohrer and N. Hogan. Avoiding spurious submovement decompositions: A globally optimal algorithm. *Biological cybernetics*, 89(3):190–199, 2003.

B. Rohrer and N. Hogan. Avoiding spurious submovement decompositions II. *Biological cybernetics*, 94 (5):409–14, 2006.

H. Rom and G. Medioni. Hierarchical decomposition and axial shape description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 973–981, 1993.

D. Rosand. *Drawing acts: Studies in graphic expression and representation.* Cambridge University Press Cambridge, 2002.

D. Rosand. Time lines. In H. De Preester, editor, *Moving imagination, explorations of gesture and inner movement in the arts*, pages 205–220. John Benjamins Publishing Company, 2013.

D. A. Rosenbaum. *Human motor control*. Academic press, 2009.

D. A. Rosenbaum, L. D. Loukopoulos, R. G. Meulenbroek, J. Vaughan, and S. E. Engelbrecht. Planning reaches by evaluating stored postures. *Psychological review*, 102(1):28, 1995.

D. A. Rosenbaum, R. G. Cohen, S. A. Jax, D. J. Weiss, and R. Van Der Wel. The problem of serial order in behavior: Lashley's legacy. *Human movement science*, 26(4):525–554, 2007.

M. E. Rosheim. *Robot evolution: The development of anthrobotics*. John Wiley & Sons, 1994.

P. Rosin. Multiscale representation and matching of curves using codons. *CVGIP: Graphical Models and Image Processing*, 55(4):286–310, Jul 1993. ISSN 1049-9652.

P. L. Rosin. Shape partitioning by convexity. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(2):202–210, 2000.

P. L. Rosin. Computing global shape measures. *Handbook of Pattern Recognition and Computer Vision*, pages 177—196, Jan 2005.

E. Roth. Graffiti taxonomy diptych: New York and Paris. http://www.evan-roth.com/work/graffiti-taxonomy-nyc-paris/, 2011.

E. Roth, T. Watson, C. Sugrue, T. Vanderlin, and J. Wilkinson. An open databse for graffiti markup language (GML) files. https://000000book.com/data, 2009.

P. K. Saha, G. Borgefors, and G. S. di Baja. A survey on skeletonization algorithms and their applications. *Pattern recognition letters*, 76:3–12, 2016.

S. Saito, A. Kani, Y. Chang, and M. Nakajima. Curvature-based stroke rendering. *The Visual Computer*, 24(1):1–11, 2008.

C. Sanderson. Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments. 2010.

T. Sanocki. Effects of font-and letter-specific experience on the perceptual processing of letters. *The American journal of psychology*, pages 435–458, 1992.

M. Sarfraz. *Interactive curve modeling*. Springer, 2008.

M. Sarfraz, A. Masood, and M. R. Asim. A new approach to corner detection. In *Computer vision and graphics*, pages 528–533. Dordrecht, 2006.

L. Scalera, E. Mazzon, P. Gallina, and A. Gasparetto. Airbrush robotic painting system: Experimental validation of a colour spray model. In *International Conference on Robotics in Alpe-Adria Danube Region*, pages 549–556. Springer, 2017.

S. Schaal. Dynamic movement primitives — a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer, 2006.

S. Schaal, P. Mohajerian, and A. Ijspeert. Dynamics systems vs. optimal control: A unifying view. *Progress in brain research*, 165:425–445, 2007.

D. Schmidlapp. *Style Writing from the Underground:(R) evolutions of Aerosol Linguistic*. IGTimes, 1996.

R. A. Schmidt. A schema theory of discrete motor skill learning. *Psychological review*, 82(4):225, 1975.

U. Schneider. A hybrid approach for stroke-based letterform composition including outline-based methods. *Computer Graphics Forum*, 19(4):243–256, 2000.

J. P. Scholz and G. Schöner. The uncontrolled manifold concept: Identifying control variables for a functional task. *Experimental Brain Research*, 126(3):289—306, 1999.

L. Schomaker. *Simulation and recognition of handwriting movements: A vertical approach to modeling human motor behavior*. PhD thesis, Katholieke Universiteit te Nijmegen, Nijmegen, The Netherlands, 1991.

L. Schomaker. A neural oscillator-network model of temporal pattern generation. *Human movement science*, 11(1):181–192, 1992.

M. Schuster. Better generative models for sequential data problems: Bidirectional recurrent mixture density networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 589–595. MIT Press, 2000.

H. S. Seah, Z. Wu, F. Tian, X. Xiao, and B. Xie. Artistic brushstroke representation and animation with disk B-spline curve. In *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pages 88—93, 2005.

W. P. Seeley. Movement, gesture, and meaning. In H. De Preester, editor, *Moving imagination, explorations of gesture and inner movement in the arts*, pages 51—68. John Benjamins Publishing Company, 2013.

D. Shaked and A. M. Bruckstein. Pruning medial axes. *Computer vision and image understanding*, 69 (2):156–169, 1998.

A. Shamir. Constraint-based approach for automatic hinting of digital typefaces. *ACM Transactions on Graphics (TOG)*, 22(2):131–151, 2003.

A. Shamir and A. Rappoport. Extraction of typographic elements from outline representations of fonts. *Computer Graphics Forum*, 15(3):259–268, 1996.

R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820): 1317–1323, 1987.

E. C. Sherbrooke, N. M. Patrikalakis, and F.-E. Wolter. Differential and topological properties of medial axis transforms. *Graphical Models and Image Processing*, 58(6):574–592, 1996.

H. Shinoda, H. Fujioka, and H. Kano. Generation of cursive characters using minimum jerk model. In *IEEE Proc. SICE*, volume 1, pages 730–3, 2003.

B. Shneiderman. Creativity support tools: A grand challenge for HCI researchers. In M. Redondo, C. Bravo, and M. Ortega, editors, *Engineering the User Interface*, pages 1–9. Springer London, 2009.

K. Shoemake. ARCBALL: a user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface*, volume 92, pages 151–156, 1992.

K. Siddiqi and B. B. Kimia. Parts of visual form: Computational aspects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):239–251, 1995.

K. Siddiqi and S. Pizer, editors. *Medial Representations: Mathematics, Algorithms and Applications*, volume 37 of *Computational Imaging and Vision series*. Springer, 2008.

M. Singh. Visual representation of contour and shape. *Oxford handbook of perceptual organization*, pages 236–258, 2015.

M. Singh and D. D. Hoffman. Part-based representations of visual shape and implications for visual cognition. In *Advances in Psychology*, volume 130, pages 401–459. 2001.

M. Singh, G. D. Seyranian, and D. D. Hoffman. Parsing silhouettes: The short-cut rule. *Perception and Psychophysics*, 61(4):636–660, 1999.

J. G. Snodgrass and M. Vanderwart. A standardized set of 260 pictures: norms for name agreement, image agreement, familiarity, and visual complexity. *Journal of experimental psychology: Human learning and memory*, 6(2):174, 1980.

R. Sosnik, B. Hauptmann, A. Karni, and T. Flash. When practice leads to co-articulation: The evolution of geometrically defined movement primitives. *Experimental Brain Research*, 156(4):422–438, 2004.

P. Spröte, F. Schmidt, and R. W. Fleming. Visual perception of shape altered by inferred causal history. *Scientific Reports*, 6(36245), 2016.

O. Stettiner and D. Chazan. A statistical parametric model for recognition and synthesis of handwriting. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, volume 2, pages 34–38. IEEE, 1994.

G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. *Information processing*, 71(1460-1465), 1972.

G. C. Stowers and P. Goldman. Graffiti art: An essay concerning the recognition of some forms of graffiti as art. *Unpublished essay, Fall*, 1997.

S. Strassmann. Hairy brushes. *ACM Siggraph Computer Graphics*, 20(4):225–232, 1986.

F. Stulp and O. Sigaud. Many regression algorithms, one unified model: A review. *Neural Networks*, 69: 60–79, 2015.

S. L. Su, Y.-Q. Xu, H.-Y. Shum, and F. Chen. Simulating artistic brushstrokes using interval splines. In *Proceedings of the 5th IASTED International Conference on Computer Graphics and Imaging*, pages 85–90, 2002.

Y. Sun, H. Qian, and Y. Xu. A geometric approach to stroke extraction for the Chinese calligraphy robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3207–3212, 2014.

I. Sutskever. *Training Recurrent Neural Networks*. PhD thesis, University of Toronto, 2013.

R. Suveeranont and T. Igarashi. Example-based automatic font generation. In *Smart Graphics*, number LNCS 6133 in Lecture Notes in Computer Science, pages 127–138. 2010.

A. Tagliasacchi. Skeletal representations and applications. Technical report, School of Computing Science, Simon Fraser University, SFU-CMPT TR 2012-55-1, 2013.

H. Tanaka, M. Tai, and N. Qian. Different predictions by the minimum variance and minimum torque-change models on the skewness of movement velocity profiles. *Neural computation*, 16(10):2021–2040, 2004.

F. Tang, W. Dong, Y. Meng, X. Mei, F. Huang, X. Zhang, and O. Deussen. Animated construction of Chinese brush paintings. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3019–3031, 2017.

S. Tang, Z. Xia, Z. Lian, Y. Tang, and J. Xiao. FontRNN: Generating Large-scale Chinese Fonts via Recurrent Neural Network. *Computer Graphics Forum*, 38(7):567–577, 2019.

A. K. Tanwani and S. Calinon. Learning robot manipulation tasks with task-parameterized semitied hidden semi-Markov model. *IEEE Robotics and Automation Letters*, 1(1):235–242, 2016.

E. Taub and A. Berman. Movement and learning in the absence of sensory feedback. *The neuropsychology of spatially oriented behavior*, 2:173–192, 1968.

C.-H. Teh and R. T. Chin. On the detection of dominant points on digital curves. *IEEE Transactions on pattern analysis and machine intelligence*, 11(8):859–872, 1989.

A. Telea. Feature preserving smoothing of shapes using saliency skeletons. In *Visualization in Medicine and Life Sciences II*, pages 153–170. Springer, 2012.

Tempt1, E. Roth, C. Sugrue, Z. Lieberman, T. Watson, and J. Powderly. The eyewriter. http://www.eyewriter.org/, 2009.

J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000.

H.-L. Teulings and L. Schomaker. Invariant properties between stroke features in handwriting. *Acta psychologica*, 82(1):69–88, 1993.

H.-L. Teulings, P. A. Mullins, and G. E. Stelmach. The elementary units of programming in handwriting. *Advances in Psychology*, 37:21–32, 1986.

Y. Thiel, K. Singh, and R. Balakrishnan. Elasticurves: Exploiting stroke dynamics and inertia for the real-time neatening of sketched 2D curves. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 383–392. ACM, 2011.

A. Thomassen and H.-L. Teulings. Time, size and shape in handwriting: Exploring spatio-temporal relationships at different levels. In J. Michon and J. Jackson, editors, *Time, Mind, and Behavior*, pages 253–263. Springer, 1985.

C. Thompson. Automated calligraphy using dynamics. Technical report, University of Washington, Dept. of Computer Science and Engineering, 2010.

S. Todd and W. Latham. *Evolutionary Art and Computers*. Academic Press, 1992.

E. Todorov. Optimality principles in sensorimotor control. *Nature neuroscience*, 7(9):907–915, 2004.

E. Todorov and M. I. Jordan. Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *Journal of Neurophysiology*, 80(2):696–714, 1998.

E. Todorov and M. I. Jordan. A minimal intervention principle for coordinated movement. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 27—34, Cambridge, MA, USA, 2002a. MIT Press.

E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5(11):1226–1235, 2002b.

J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley. Procedural content generation: Goals, challenges and actionable steps. In *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 61–75. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

E. B. Torres, R. Quian Quiroga, H. Cui, and C. Buneo. Neural correlates of learning and trajectory planning in the posterior parietal cortex. *Frontiers in integrative neuroscience*, 7:39, 2013.

A. Treisman and S. Gormican. Feature analysis in early vision: Evidence from search asymmetries. *Psychological Review*, 95(1):15–48, 1988.

P. Tresset and F. Fol Leymarie. Portrait drawing by Paul the robot. *Computers & Graphics*, 37(5):348–63, 2013.

M. Turvey, H. L. Fitch, and B. Tuller. The Bernstein perspective: I. The problems of degrees of freedom and context-conditioned variability. *Human motor behavior: An introduction*, pages 239–252, 1982.

S. Ullman. Filling-in the gaps: The shape of subjective contours and a model for their generation. *Biological Cybernetics*, 25(1):1–6, 1976.

M. A. Umilta, C. Berchio, M. Sestito, D. Freedberg, and V. Gallese. Abstract art and cortical motor activation: An EEG study. *Frontiers in human neuroscience*, 6, 2012.

Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological cybernetics*, 61(2):89–101, 1989.

J. D. van der Gon, J. P. Thuring, and J. Strackee. A handwriting simulator. *Physics in medicine and Biology*, 6(3):407–14, 1962.

S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

R. R. van Doorn and P. J. Keuss. Does the production of letter strokes in handwriting benefit from vision? *Acta Psychologica*, 82(1–3):275–290, 1993.

A. S. Vempati, M. Kamel, N. Stilinovic, Q. Zhang, D. Reusser, I. Sa, J. Nieto, R. Siegwart, and P. Beardsley. PaintCopter: An autonomous UAV for spray painting on three-dimensional surfaces. *IEEE Robotics and Automation Letters*, 3(4):2862–2869, 2018.

P. Viviani and T. Flash. Minimum-jerk, two-thirds power law, and isochrony: Converging approaches to movement planning. *Journal of Experimental Psychology: Human Perception and Performance*, 21 (1):32, 1995.

P. Viviani and G. McCollum. The relation between linear extent and velocity in drawing movements. *Neuroscience*, page 211218, 1983.

P. Viviani and R. Schneider. A developmental study of the relationship between geometry and kinematics in drawing movements. *Journal of Experimental Psychology: Human Perception and Performance*, 17(1):198, 1991.

P. Viviani and C. Terzuolo. Trajectory determines movement dynamics. *Neuroscience*, 7(2):431–437, 1982.

J. Vredenbregt and W. Koster. Analysis and synthesis of handwriting. *Philips Technical Review*, 32(3): 73–78, 1971.

Y. Wada and M. Kawato. A theory for cursive handwriting based on the minimization principle. *Biological Cybernetics*, 73(1):3–13, 1995.

J. Wagemans. Perceptual organization. In *Stevens' Handbook of Experimental Psychology and Cognitive Neuroscience, Sensation, Perception, and Attention*, volume 2, chapter 18, pages 803–872. John Wiley & Sons, 2018. 4th Edition.

J. Wagemans, A. J. van Doorn, and J. J. Koenderink. Measuring 3D point configurations in pictorial space. *i-Perception*, 2(1):77–111, 2011.

J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt. A century of Gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological Bulletin*, 138(6):1172–1217, 2012.

D. Walton and D. Meek. An improved Euler spiral algorithm for shape completion. In *Computer and Robot Vision, 2008. CRV '08. Canadian Conference on*, pages 237–244, May 2008.

Y. Wamain, V. Kostrubiec, M. Longcamp, J. Tallet, and P. G. Zanone. Does graphic shapes perception mirror handwriting patterns production? *Advances in Graphonomics: Proceedings of IGS 2009*, pages 202–205, 2009.

Y. Wamain, J. Tallet, P.-G. Zanone, and M. Longcamp. Brain responses to handwritten and printed letters differentially depend on the activation state of the primary motor cortex. *Neuroimage*, 63(3): 1766–1773, 2012.

C. Wang and Z. Lai. Shape decomposition and classification by searching optimal part pruning sequence. *Pattern Recognition*, 54(October):206–217, 2016.

J. Wang, C. Wu, Y.-Q. Xu, H.-Y. Shum, and L. Ji. Learning-based cursive handwriting synthesis. In *Eighth IEEE International Workshop on Frontiers in Handwriting Recognition*, pages 157–162, 2002.

J. Wang, C. Wu, Y.-Q. Xu, and H.-Y. Shum. Combining shape and physical modelsfor online cursive handwriting synthesis. *International Journal of Document Analysis and Recognition (IJDAR)*, 7(4): 219–227, 2005.

X. Wang, X. Liang, L. Sun, and M. Liu. Triangular mesh-based stroke segmentation for Chinese calligraphy. In *12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1155–1159, 2013.

Y. Wang. Interview with Charles Bigelow. *TUGboat*, 34(2):136–167, 2013.

Y. Wang, Y. Gao, and Z. Lian. Attribute2Font: Creating fonts you want from attributes. *arXiv preprint arXiv:2005.07865*, 2020.

W. C. Watt. Canons of alphabetic change. In *The alphabet and the brain*, pages 122–152. Springer, 1988.

D.-L. Way, W.-J. Lin, and Z.-C. Shih. Computer-generated Chinese color ink paintings. *Journal of the Chinese Institute of Engineers*, 29(6):1041–1050, 2006.

C. Wen, J. Chang, Y. Zhang, S. Chen, Y. Wang, M. Han, and Q. Tian. Handwritten Chinese font generation with collaborative stroke refinement. *arXiv preprint arXiv:1904.13268*, 2019.

M. Wertheimer. Untersuchungen zur lehre von der gestalt. ii. *Psychologische forschung*, 4(1):301–350, 1923.

C.-F. Westin, S. E. Maier, H. Mamata, A. Nabavi, F. A. Jolesz, and R. Kikinis. Processing and visualization for diffusion tensor MRI. *Medical Image Analysis*, 6(2):93–108, 2002.

S. Wiewel, M. Becher, and N. Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. *Computer Graphics Forum*, 38(2):71–82, 2019.

K. Wiley and L. R. Williams. Representation of interwoven surfaces in 2 1/2 D drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 65–74. ACM, 2006.

L. Williams and K. K. Thornber. Orientation, scale, and discontinuity as emergent properties of illusory contour shape. *Neural Computation*, 13(8):1683–1711, 2001.

L. R. Williams and D. W. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. *Neural Computation*, 9(4):837—858, 1997.

A. Witkin. Scale-space filtering. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1019—22, Karlsruhe, West Germany, August 1983.

A. Woch and R. Plamondon. The problem of movement primitives in the context of the kinematic theory. In *Proceeding of the 11th Conference of the International Graphonomics Society*, pages 67–71, 2003.

D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan. Principles of sensorimotor learning. *Nature Reviews Neuroscience*, 12(12):739–751, 2011.

F.-E. Wolter. Cut locus and medial axis in global shape interrogation and representation. Ocean engineering design laboratory memorandum 92-2, MIT, Cambridge, MA, USA, 1992.

F.-E. Wolter and K.-I. Friese. Local and global geometric methods for analysis interrogation, reconstruction, modification and design of shape. In *IEEE Proceedings of the International Conference on Computer Graphics (CGI)*, pages 137–152, June 2000.

A. L. Wong, J. Goldsmith, and J. W. Krakauer. A motor planning stage represents the shape of upcoming movement trajectories. *Journal of neurophysiology*, 116(2):296–305, 2016.

R. S. Woodworth. Accuracy of voluntary movement. *The Psychological Review: Monograph Supplements*, 3(3):i, 1899.

J. Xu and C. S. Kaplan. Calligraphic packing. In *Proceedings of Graphics Interface 2007*, pages 43–50, 2007.

S. Xu, F. C. Lau, and Y. Pan. *A Computational Approach to Digital Chinese Painting and Calligraphy*. Springer, 2009.

S. Xu, H. Jiang, F. C. Lau, and Y. Pan. Computationally evaluating and reproducing the beauty of Chinese calligraphy. *IEEE Intelligent Systems*, 27(3):63–72, 2012.

Y. Xu and M. Singh. Early computation of part structure: Evidence from visual search. *Perception and Psychophysics*, 64(7):1039–1054, 2002.

Z. Yan, S. Schiller, G. Wilensky, N. Carr, and S. Schaefer. k-curves: Interpolation at local maximum curvature. *ACM Transactions on Graphics (TOG)*, 36(4):129, 2017.

S. C. Yen and L. H. Finkel. Extraction of perceptually salient contours by striate cortical networks. *Vision Research*, 38(5):719–741, 1998.

J. Yu and Q. Peng. Realistic synthesis of *cao shu* of Chinese calligraphy. *Computers & Graphics*, 29(1): 145–153, 2005.

M. E. Yumer, P. Asente, R. Mech, and L. B. Kara. Procedural modeling using autoencoder networks. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*, pages 109–118, 2015.

W. Zaner-Bloser method. Zaner-Bloser — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Zaner-Bloser`, 2020. [Online; accessed 13-December-2020].

M. Zeestraten, S. Calinon, and D. G. Caldwell. Variable duration movement encoding with minimal intervention control. In *Proc. of the IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 497–503, May 2016a.

M. J. A. Zeestraten, S. Calinon, and D. G. Caldwell. Variable duration movement encoding with minimal intervention control. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 497–503, May 2016b.

D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern recognition*, 37(1):1–19, 2004.

J. Zhang, Y. Wang, W. Xiao, and Z. Luo. Synthesizing ornamental typefaces. *Computer Graphics Forum*, 36(1):64–75, 2017a.

X. Zhang and G. Liu. Chinese calligraphy character image synthesis based on retrieval. In *Advances in Multimedia Information Processing-PCM 2009*, pages 167–178. Springer, 2009.

X.-Y. Zhang, F. Yin, Y.-M. Zhang, C.-L. Liu, and Y. Bengio. Drawing and recognizing Chinese characters with recurrent neural network. *IEEE transactions on pattern analysis and machine intelligence*, 40 (4):849–862, 2017b.

Z. Zhang, J. Tomlinson, and C. Martin. Splines and linear control theory. *Acta Applicandae Mathematica*, 49(1):1–34, 1997.

C. L. Zitnick. Handwriting beautification using token means. *ACM Transactions on Graphics (TOG)*, 32 (4):53, 2013.

C. Zou, J. Cao, W. Ranaweera, I. Alhashim, P. Tan, A. Sheffer, and H. Zhang. Legible compact calligrams. *ACM Transactions on Graphics (TOG)*, 35(4), 2016. Article no. 122.

L. Zusne. *Visual perception of form*. Academic Press New York, 1970.