Computational Aesthetics in Graphics, Visualization, and Imaging
Paul Rosin (Editor)

Computational Models for the Analysis and Synthesis of Graffiti Tag Strokes

Daniel Berio[†1] and Frederic Fol Leymarie[‡1]

[1]Goldsmiths College, University of London

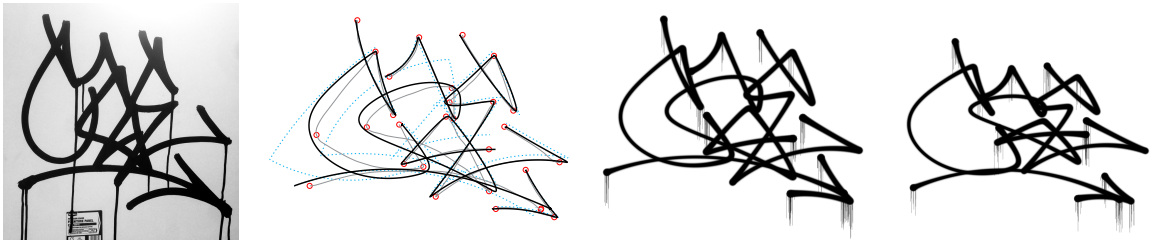

Figure 1: *From walls to digital. A graffiti tag (CAE) made with an ink marker. Right, the system described in this paper in action: (1) Trajectory (black) reconstructed from an input trace made with a trackpad (light grey) and the corresponding "action plan" (dotted turquoise). (2) Rendering of the tag. (3) Rendering of the tag with modified model parameters.*

Abstract

In this paper we describe a system aimed at the generation and analysis of graffiti tags. We argue that the dynamics of the movement involved in generating tags is in large part — and at a higher degree with respect to many other visual art forms — determinant of their stylistic quality. To capture this notion computationally, we rely on a biophysically plausible model of handwriting gestures (the Sigma Lognormal Model proposed by Réjean Plamondon et al.) that permits the generation of curves which are aesthetically and kinetically similar to the ones made by a human hand when writing. We build upon this model and extend it in order to facilitate the interactive construction and manipulation of digital tags. We then describe a method that reconstructs any planar curve or a sequence of planar points with a set of corresponding model parameters. By doing so, we seek to recover plausible velocity and temporal information for a static trace. We present a number of applications of our system: (i) the interactive design of curves that closely resemble the ones typically observed in graffiti art; (ii) the stylisation and beautification of input point sequences via curves that evoke a smooth and rapidly executed movement; (iii) the generation of multiple instances of a synthetic tag from a single example. This last application is a step in the direction of our longer term plan of realising a system which is capable of automatically generating convincing images in the graffiti style space.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

The word graffiti is generally used to indicate any form of drawing made without permission on public surfaces. In this paper we use the term referring to an art movement, also known as *writing*, that originated in the late 1960s on the sur-

† d.berio@gold.ac.uk
‡ ffl@gold.ac.uk

faces of the New York Subway, and which revolves around the abstraction and stylisation of the letters of an artist's pseudonym [SG97, Kim14]. Our work draws on the personal experience of the first author as a graffiti artist and is aimed at the computational generation and analysis of forms in the graffiti style space (Figure 1).

In this study we focus on the most basic form of graffiti, commonly referred to as *tags*: highly stylised signatures that convey the artist's identity, style and skill (Figure 2). The letters of a tag are usually not important in their meaning as a word, but rather are meant to impress by their visual quality. Tags are meant to be executed quickly and in great quantity; the speed and spontaneity of execution is determinant in the genesis of their shape and dynamics. With experience, tags are written more rapidly and the gestures involved in their creation are interiorised, resulting in more fluid movements and spontaneous forms. We follow the hypothesis that an experienced artist's hand will draw more efficient curves [GG10], which will ultimately result in more harmonious and aesthetically pleasing forms. As such, one of the determinant factors used by graffiti writers when aesthetically judging a tag is its "flow", *i.e.* the confidence and smoothness of the motion with which it has been executed. The implicit assumption is that such a flow can be recovered from the static traces of a finished art work.

Figure 2: *An example of the aesthetic we try to achieve in this study. Tags by graffiti artists Geso and Nemel (source, graffuturism.com).*

Various studies suggest that the recovery of motion from a static trace is key to its perception. Lacquaniti *et al.* [LTV83] have observed a power law relating the curvature of a hand trajectory to the angular velocity of the movement that generated it (the *2/3 power law*). Further studies also relate tangential velocity to curvature [VM83] and identify the correspondence between units of action and landmark points along the generated trajectory [VC85]. These relationships between figural and dynamic aspects of a drawing point to a, possibly innate ability of a human to recover a motion by observing a graphic sign [VS92]. Neuroscientists have suggested that areas of the brain that correspond with motor control are activated during the observation of static traces [FG07] and in particular, when viewing instances of handwriting [LARV03]. We hypothesize that the mental reconstruction of motion from a static trace is of fundamental importance in the perception and aesthetic evaluation of graffiti tags.

For the task of capturing computationally the stylistic qualities of tags, and evoking in the viewer the kind of gestures that are typically employed in their genesis, it is convenient to rely on a method that does not only describe the shape of the trace but also the velocity of the movement that generates it. To do so, we rely on a family of bio-physically plausible models of handwriting, the *Kinematic Theory of Rapid Human Movements* [Pla95], which is well known in the handwriting analysis and synthesis domains. We employ these models in order to generate curves which are aesthetically and kinetically similar to the ones that would be made by an expert graffiti artist.

The paper is organised as follows: After an overview of related works and background, we summarise Plamondon et al.'s Kinematic Theory of Rapid Human Movements and the Sigma Lognormal ($\Sigma\Lambda$) model [Pla95], which form the basis for our tag generation model (*Section 3*). In *Section 4* we extend the Kinematic Theory and build an Euler spiral based trajectory model ($\Gamma\Lambda$) that allows to describe trajectories that contain inflections with a reduced number of parameters. For both the $\Sigma\Lambda$ and $\Gamma\Lambda$ models, we describe intermediate representations ($\Sigma\Lambda^*$ and $\Gamma\Lambda^*$) aimed at facilitating the interactive specification and manipulation of trajectories. In *Section 5* we demonstrate how the $\Sigma\Lambda$ and $\Gamma\Lambda^*$ models can be used as a tool for the computer aided design, rendering and animation of realistic digital tags. In *Section 6* we describe a method that reconstructs a static trace (given as a sequence of planar points) with $\Gamma\Lambda^*$ parameters. We then show how the same technique can be used to reconstruct $\Sigma\Lambda^*$ by means of a small modification. Finally, in *Section 7* we describe how our system can be used to generate multiple instances of a synthetic tag from a single example.

Throughout the paper we rely on the $\Sigma\Lambda^*$ and $\Gamma\Lambda^*$ reparameterisations. Nevertheless, it should be noted that the underlying models remain identical, and thus the reparameterisations are interchangeable with the original model. Furthermore, we consider it is important to emphasise the applicability of the techniques described in this paper for both the $\Sigma\Lambda$ and the $\Gamma\Lambda$ models. In fact, while the $\Gamma\Lambda$ model is advantageous for the interactive specification and fitting of trajectories, we do not claim its biological plausibility. On the other hand, the $\Sigma\Lambda$ model has been widely used in the handwriting synthesis and analysis domains, therefore its application expands the scope of our work to the broader context of handwriting analysis and synthesis.

2. Background

2.1. Graffiti in the Computational Domain

Previous work has been done at the intersection of graffiti art with technology. Jurg Lehni has developed HEKTOR, a Cartesian drawing machine that is able to mechanically trace vector images with a spray can on a wall [Leh04]. Evan Roth initiated the Graffiti Analysis project which has resulted in a series of low cost motion capture devices that allow to easily record the gestures done during tagging [Rot04]. A large online database stores the motion data recorded with such devices and we use this data for an experiment in the present study (§6.6). The EyeWriter project [eye09] is an eye tracking based system that has allowed Tempt One, a graffiti artist affected by a degenerative nerve disorder, to practice his art despite being paralysed. Kanno and Yamaguchi have developed the Senseless Drawing Bot [KY12], an installation where a wheeled robot uses a double pendulum to draw chaotic traces with spray paint on a wall. This last project is particularly interesting from a generative standpoint; in fact the rapid swinging motion of the pendulum creates trajectories that are similar to the ones that can be seen in tags produced by the human hand. To the best of our knowledge our work is the first in this domain to systematically study methods for the computational synthesis and analysis of graffiti art.

2.2. Curve Fairing and Sketch Based Systems

The technique described in this communication is related to a series of works in the fields of (i) curve *fairing*, *i.e.* the process of removing imperfections or generating smooth curvature profiles for a noisy input curve, and (ii) *stylisation and beautification*, *i.e.* the process of generating curves with a specific style from a noisy or approximated input. In our method we 'beautify' and stylise a user input by simulating the rapid motion of an expert hand. Thiel *et al.* [TSB11] interactively beautify 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 motions when they intend to create a more precise drawing. Lu *et al.* [LYFD12] stylise an input point sequence by adaptively fitting examples made by expert artists on tablets with a high number of degrees of freedom (*i.e.* pen tilt/pressure). The system reconstructs pen tilt and pressure information from the input, thus allowing the creation of more realistic brush renderings. Our work is similar in that we "hallucinate" information that is missing (or not considered) in the input – in our case velocity. Inspired by texture synthesis works, Hertzmann *et al.* [HOCS02] describe a data driven approach to stylise an input stroke in different styles by finding mappings to an example database. Zitnick [Zit13] fairs 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. Xie *et al.* [XHLW14] build a system that acts as an *assistant* in the process of drawing digital portraits from a bitmap image. Outline strokes are smoothed and adapted to salient features of the image (e.g. edges) and shading strokes are adjusted to better match the luminosity of the input. Our work is related in spirit, as we aim at building a system that assists the designer in the process of specifying graffiti tags. AlMeraj *et al.* [AWI*09] develop a system that mimics the visual character of hand-drawn pencil lines. Similarly to the present study, pencil traces are generated by applying a physical model of hand-writing movements [FH85]. On the other hand, the physical model is used to mimic the high frequency features (e.g the undulation of drawn lines) that can be seen in drawings made with a pencil. In our case, we apply a physical model at a higher level to describe the overall curvature and form of the simulated graffiti trace. McCrae and Singh [MS09] fair an input contour by fitting and interpolating a minimum number of clothoid (Euler spiral) curves to parts of the contour with a linear curvature variation. Havemann *et al.* [HEWF13] use a discrete approximation of clothoid curves as an alternative to splines or Bézier for computer aided applications. In Section 6 we also take advantage of the descriptive power of clothoids to extend our trajectory generation model.

2.3. Graphonomics

Graphonomics is the field directed at the scientific study of handwriting and other related graphic skills [KHVG86]. Through the years, various models have been proposed to describe the velocity, curvature and other features of handwriting movements.

Hollerbach [Hol81] describes handwriting with the phase and amplitude modulation of superimposed horizontal and vertical oscillations. Flash and Hogan [FH85] describe the point to point movements in handwriting with a quintic equation that minimises jerk (*i.e.* the derivative of acceleration). Various models of handwriting rely on the notion that movements are generated by the superimposition in time of multiple simple movement primitives or "ballistic strokes" [TS93]. Morasso and Mussa Ivaldi [MMI82] synthesise handwriting traces by using a weighted sum over time of strokes with a shape defined by B-Splines. Bullock *et al.* [BGM93] simulate neural signals to generate smooth handwriting trajectories that interpolate a motor plan made by a sequence of positions. Rejan Plamondon has developed the Kinematic Theory of Rapid Human Movements [Pla95] which describes handwriting movements as the sum in time of stroke primitives with an asymmetric bell-shaped velocity profile that is described with a log-normal function (1). Similarly, Bezine *et al.* [BAS04] use stroke primitives with an elliptic form and an asymmetric velocity profile characterised by a Beta function.

Li *et al.* [LPP98] use circular arcs and line segments to

compress a handwriting trace by segmenting it at curvature extrema and inflection points. The segmentation process is similar to ours, but the kinetic aspects are not taken into account. For the purpose of training handwriting classifiers, Varga *et al.* [VKB05] generate various samples of synthetic handwriting by using a set of Bézier curve templates of handwritten characters. Handwriting trajectories are generated by fitting a model from the Kinematic Theory to the input curve and varying its parameters. Also in this case the parameters for the model are determined by segmenting the input curve at curvature extrema, but the segmentation is computed analytically given the parametric definition of the character templates.

3. Kinematic Theory of Rapid Human Movements

We have chosen to build our graffiti generation system around Plamondon's Kinematic Theory. This decision was initially based on the first author's impression that the velocity of his own movement while drawing could be described with an asymmetric velocity curve. Our choice was further encouraged by experimental results achieved during the initial research phases, that have shown that the application of the Kinematic Theory indeed permits the generation of curves that show a strong resemblance to the ones that can be seen in graffiti art, and further that are similar to the ones executed by an experienced drawing hand. Plamondon suggests that his theory is aimed at describing ideally *well learned* handwriting movements [PORD13]. This concept is in accord with the hypothesis that the experienced artist, after years of practice, will be capable of synthesing effortlessly aesthetically pleasing and distinct traces.

The Kinematic Theory describes handwriting gestures with the vectorial sum in time of elementary movement primitives denoted as *strokes*. The velocity of each stroke results from the impulse response of a large number of coupled neural and muscular subsystems to an activation command by the central nervous system. Plamondon proves mathematically that such impulse response asymptotically converges to a lognormal curve [PFW03]. Thus the velocity profile of a stroke will assume a variably asymmetric 'bell shape' which is given by the following equation:

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

where t_0 is the time of occurrence of the input command for the stroke and (μ,σ) determine the overall shape of the lognormal. μ is the stroke time-delay in logarithmic time scale (logtime delay) and indicates the rapidity of the system to react to the input command; σ is the stroke response-time in a logarithmic time scale (logresponse time) and determines the spread and asymmetry of the lognormal. For an in depth discussion of the effects and meanings of the lognormal parameters we refer the reader to Plamondon, Feng & Woch [PFW03].

3.1. Sigma Lognormal Model ($\Sigma\Lambda$)

The Kinematic Theory comprises a framework of different models capable of describing gestures of varying complexity, ranging from fast reaching movements to complex handwriting trajectories [PD06]. The most flexible and descriptive of these models is the Sigma Lognormal Model ($\Sigma\Lambda$ Model) which allows the definition of curved trajectories with a circle-arc based stroke primitive. The arc is used with the assumption that single sub-movements are made around a pivot point [PDO09]. Each stroke S_j is described by a set of parameters:

$$S_j = \{t_{0j}, \mu_j, \sigma_j, \theta_{1j}, \theta_{2j}, D_j\}, \quad (2)$$

where t_{0j}, μ_j and σ_j are the same parameters used in (1), θ_{1j} and θ_{2j} are respectively the initial and final angular deviation of the stroke, while D_j is the amplitude of the stroke command (*i.e* the distance to be covered).

The instantaneous speed of a single stroke at time t is defined by:

$$v_j(t) = D_j\Lambda(t), \quad (3)$$

the direction of the stroke is interpolated between θ_{1j} and θ_{2j} via a sigmoid function (erf) with the following equation:

$$\omega_j(t) = \theta_{1j} + \frac{\theta_{2j}-\theta_{1j}}{2}\left(1 - \mathrm{erf}\left(\frac{\ln(t-t_{0j})-\mu_j}{\sigma_j\sqrt{2}}\right)\right). \quad (4)$$

The planar position of the effector/pen-tip at time t is then given by:

$$p(t) = p_{start} + \int_0^t \left(\sum_{j=1}^{n} v_j(\tau)\begin{bmatrix}\cos\omega_j(\tau)\\ \sin\omega_j(\tau)\end{bmatrix}\right) d\tau, \quad (5)$$

where $\boldsymbol{p}_{start}$ is the starting point of the trajectory.

The $\Sigma\Lambda$ model describes trajectories with an *action plan* made up of a set of *virtual targets*, *i.e.* a set of imaginary way-points that define the evolution of the trajectory in time (Figure 3.a). A stroke aimed toward the next target may begin earlier than the end of the current active stroke, resulting in an overlap between the respective velocity profiles. A greater time overlap will result in a smoother trajectory (Figure 3.b).

3.2. $\Sigma\Lambda^\star$ Reparametrisation

The direct specification of $\Sigma\Lambda$ parameters poses some problems when used in a design-oriented task. As an example, each virtual target position is defined as a relative offset with respect to the preceding target. This results in a low degree of *locality* [Knu79], *i.e.* the variation of certain stroke parameters at the beginning of a trajectory, will greatly influence the final shape of the generated curve.

To overcome this limitation and to simplify the user definition and manipulation of trajectories in a computer aided

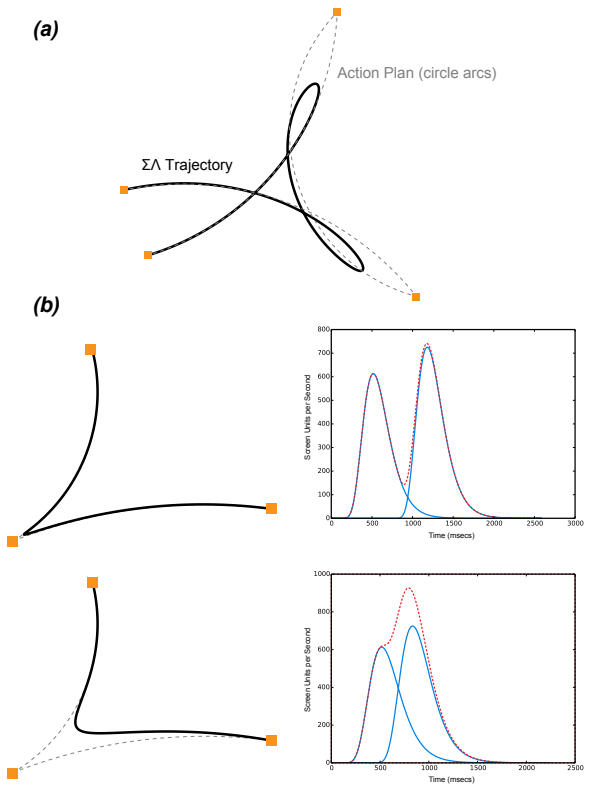

Figure 3: ΣΛ *Model.* ***(a)*** *The action plan that defines a complex trajectory.* ***(b)*** *Smoothing trajectories by varying the time overlap of the stroke velocity profiles (dashed red: the trajectory velocity profile. Turquoise: the single stroke velocity profiles.*

design context, we specify ΣΛ trajectories with an *intermediate representation* of its parameters, here denoted as $\Sigma\Lambda^{\star}$. We describe a $\Sigma\Lambda^{\star}$ trajectory with an action plan made of m explicitly defined virtual target positions $(\boldsymbol{w}_j, 0 \leq j < m)$ and a series of $m-1$ strokes. The initial position of the trajectory is given by $\boldsymbol{w}_0$, and each stroke $S_j^{\star}$ is defined with the following parameters:

$$S_j^{*} = \{\Delta t_j, \mu_j, \sigma_j, \phi_j\}, \tag{6}$$

where Δt_j is the time offset relative to the previous stroke S_{j-1} (0 for $j=1$), μ_j and σ_j are the log-delay and log-response time, and ϕ_j is an angle which defines the curvature with respect to the stroke's principal direction $\Delta\boldsymbol{w} = \boldsymbol{w}_j - \boldsymbol{w}_{j-1}$. Given a series of virtual targets and $\Sigma\Lambda^{*}$ stroke parameters we can easily compute the corresponding ΣΛ trajectory parameters with the following relations:

$$t_{0j} = t_{0j-1} + \Delta t_{j-1}, \tag{7}$$

$$\theta_{1j} = \tan^{-1}\left(\frac{\Delta \boldsymbol{w}_y}{\Delta \boldsymbol{w}_x}\right) - \phi_j, \tag{8}$$

$$\theta_{2j} = \tan^{-1}\left(\frac{\Delta \boldsymbol{w}_y}{\Delta \boldsymbol{w}_x}\right) + \phi_j, \tag{9}$$

$$D_j = 2\,\|\Delta\boldsymbol{w}\|\sin(\phi_j), \tag{10}$$

4. Gamma Lognormal Model (ΓΛ)

Using the $\Sigma\Lambda^{\star}$ facilitates the interactive definition of trajectories. However, we have noted that it is difficult to describe trajectories that contain *inflection points*. For example, consider the looping form of the numeral "8" (Figure 4, left). While it is indeed possible to generate this type of trajectory by adding virtual targets at inflections, the placement of such points will not result in an intuitive process. To overcome this limitation, we have developed an extension to the ΣΛ model, the Gamma Lognormal Model (ΓΛ), that uses a stroke primitive which has a higher descriptive power and allows the definition of inflections at the single-stroke level by using an Euler spiral (Figure 4, right). We choose to use the greek letter Γ referring to Graffiti (Γκράφιτι) and because the equation of the Euler spiral is related to the Gamma function [Lev08]. The proposed model offers more flexibility and with a few constraints it is also able to generate the exact same trajectories as the ones generated by the ΣΛ model, since the same circle-arc strokes can still be described with the Euler spiral primitive. In the rest of the paper we will focus on applications of this extended model.

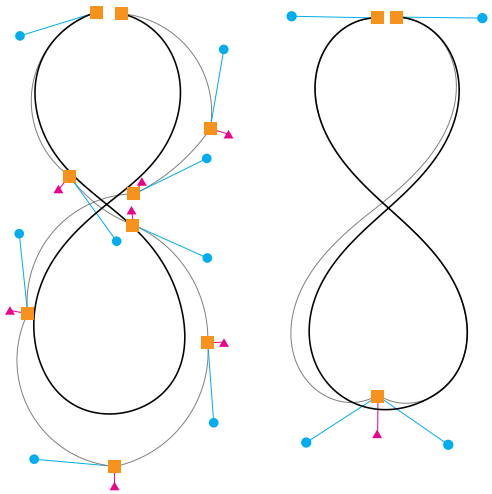

Figure 4: *A comparision between* $\Sigma\Lambda^{\star}$ *(left) and* $\Gamma\Lambda^{\star}$ *(right) models in the task of describing a "figure of 8". In orange (■): the virtual target positions. In turquoise (●): the handles used for manipulating the* ϕ_j *parameters (left) or the* ϕ_{1j} *and* ϕ_{2j} *parameters (right). In green (▲): the handles used for adjusting the* Δt_j *parameters.*

4.1. Euler Spirals

The (circular) arc has a constant curvature of $\frac{1}{r}$. At a descriptive level, the natural generalisation of the arc is the Euler spiral (also commonly known as Cornu's spiral, or Clothoid), a curve with curvature that is a linear function of arc length [Lev08]. Euler spirals have a rich history and are widely used in many domains, notably for the design of smooth transition paths for railway tracks, but also for curve fairing [MS09] and curve completion [KFP03, CK14]. The Euler spiral can be parametrised by arc length via the Fresnel sine and cosine integrals:

$$C(t) = \int_0^t \cos\left(\frac{\pi}{2}x^2\right) dx, \qquad S(t) = \int_0^t \sin\left(\frac{\pi}{2}x^2\right) dx. \tag{11}$$

Given two points and a pair of corresponding tangent directions (Hermite points), it is possible to calculate a spiral that will connect the two points with tangents equal to the respective Hermite constraints. There is a variety of techniques available for this calculation [KFP03, WM08, Lev09, BF13]; we currently are using the technique developed by Walton & Meek [WM08] in which an Euler spiral is fitted to a pair of Hermite points by numerically solving a small system of non-linear equations. The positions along the fitted spiral are given by:

$$\boldsymbol{e}(t) = \boldsymbol{p}_0 + \mathrm{sgn}(t)aC(t)\boldsymbol{t} + \mathrm{sgn}(t)aS(t)\boldsymbol{n}, \tag{12}$$

where $C(s)$ and $S(s)$ are the Fresnel cosine and sine integrals (11), $\boldsymbol{p}_0$ is the centre of the spiral ($s = 0$), $\boldsymbol{t}$ and $\boldsymbol{n}$ are the unit tangent and normal at $\boldsymbol{p}_0$, a is a scaling parameter and $sgn(t)$ defines the positive and negative parts of the spiral (Figure 5). For a more detailed description, we refer the reader to Walton & Meek [WM08, WM09] and Connor & Krivodonova [CK14].

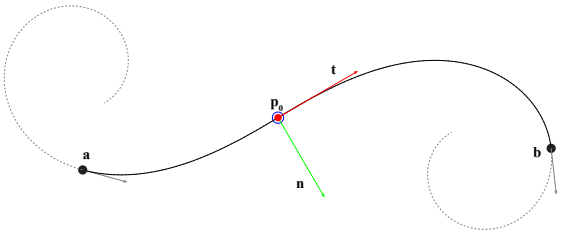

Figure 5: *Definition of an Euler spiral connecting two Hermite points $\boldsymbol{a}$ and $\boldsymbol{b}$ based on Walton and Meek's definition.*

4.2. ΓΛ Trajectories

We define a ΓΛ trajectory with a series of stroke primitives and an initial position $\boldsymbol{p}_0$. Each stroke primitive $S_{\gamma j}$ is defined by the following parameters:

$$\{t_0, \mu, \sigma, \boldsymbol{t}, \boldsymbol{n}, s_1, s_2, a\}, \tag{13}$$

where t_0, μ, σ assume the same meaning as in the ΣΛ Model (2) and $\boldsymbol{t}, \boldsymbol{n}, s_1, s_2, a$ are the Euler spiral parameters as used in Eq. 12.

Given the first derivatives of the fresnel sine and cosine functions, $C'(t) = \cos(\frac{\pi}{2}t^2)$ and $S'(t) = \sin(\frac{\pi}{2}t^2)$ we compute a ΓΛ trajectory with the following equations:

$$s_j(t) = \frac{1}{2}\left(1 - erf\left(\frac{\ln(t - t_{0j}) - \mu_j}{\sigma_j\sqrt{2}}\right)\right), \tag{14}$$

$$\boldsymbol{\omega}_{\gamma j}(t) = D_j \frac{s_{2j} - s_{1j}}{|s_{2j} - s_{1j}|}(C'(s_j(t))\boldsymbol{t} + S'(s_j(t))\boldsymbol{n}). \tag{15}$$

The (vector) planar position along the trajectory at time t is then given by:

$$\boldsymbol{p}(t) = \boldsymbol{p}_0 + \int_0^t \left(\sum_{j=1}^{n} v_j(\tau)\boldsymbol{\omega}_{\gamma j}(\tau)\right) d\tau, \tag{16}$$

where $v_i(\tau)$ is the lognormal velocity function defined in (3).

4.3. ΓΛ* Reparametrisation

Euler spiral parameters are counter-intuitive to specify manually, so similarly to the ΣΛ* case, we define a ΓΛ* reparametrisation where a trajectory is defined by an action plan made of m virtual target positions $(\boldsymbol{w}_j, 0 \leq j < m)$ and a series of $m-1$ strokes. Each stroke is defined with the following parameters:

$$S^*_{\gamma j} = \{\Delta t_j, \mu_j, \sigma_j, \phi_{1j}, \phi_{2j}\}, \tag{17}$$

where Δt_j is the time offset relative to the previous stroke $S^*_{\gamma j-1}$ (0 for $j = 1$), μ_j and σ_j are the log-delay and log-response time, and ϕ_{1j} and ϕ_{2j} are the angular deviation of the tangents defining the spiral with respect to the strokes principal direction $\boldsymbol{d} = \boldsymbol{w}_j - \boldsymbol{w}_{j-1}$. Given a series of virtual targets and ΓΛ* stroke parameters we can easily compute the corresponding ΓΛ trajectory with:

$$\boldsymbol{t}, \boldsymbol{n}, s_1, s_2, a = \mathrm{fitEuler}(\boldsymbol{w}_{j-1}, \phi_{1j}, \boldsymbol{w}_j, \phi_{2j}), \tag{18}$$

where fitEuler uses the Walton and Meek method [WM08] to fit an Euler spiral to the pair of points $\boldsymbol{w}_{j-1}$ and $\boldsymbol{w}_j$ and the respective tangent directions defined by the angular deviations ϕ_{1j} and ϕ_{2j}. Computing the t_0 parameters is trivial, with $t_{0j} = t_{0j-1} + \Delta t_{j-1}$ when $j > 1$ and $t_{1j} = 0$.

5. Interaction and Rendering

In computer graphics applications, curves are commonly generated by using polynomial interpolation, relying on techniques such as B-splines or Bézier curves. In such methods a curve segment is parametrically defined by specifying its end points and additional control points defining its curving behaviour. The manual choice of control point positions is often a tedious and counterintuitive task, especially when

specifying curves that are similar to the ones that would be drawn by a human artist's hand.

We describe a method that allows a user to design complex graffiti-like trajectories via the interactive specification of $\Gamma\Lambda^\star$ or $\Sigma\Lambda^\star$ parameters. We argue that this is a more intuitive and direct way of specifying such curves when compared to the polynomial interpolation counterpart.

The implementation of our system relies on three simplifying assumptions that facilitate user interaction and also trajectory fitting: **(1)** The duration of a stroke is kept independent of its amplitude. This assumption is based on a common notion in the study of hand-writing movements – the *isochrony principle* [VM83, TT85] – which states that movement velocity increases proportionally to its amplitude (magnitude); as a result movement time remains relatively independent from the size of a trajectory. **(2)** We consider σ and μ global parameters of the neuro-muscular system [POG*14], and keep them constant across strokes. In our experiments we have empirically set the values to $\mu = ln(0.2)$ and $\sigma = ln(1.4)$ but, in future developments, we plan to extract these parameters through the analysis of the writing motions of an artist. **(3)** We set a time range for each stroke such that Δt_j will vary between a minimum Δt_{min} and a maximum Δt_{max}.

The point and click process initially generates a trajectory where each stroke $S^*_{\gamma j}$ has initial values $\Delta t_j = \Delta t_{min} + (\Delta t_{max} - \Delta t_{min}) * 0.5$ and $\phi_{1j} = \phi_{2j} = 0$. We then let the user adjust each ϕ_{1j} and ϕ_{2j} by dragging handles rotating around pivot points respectively defined at the corresponding virtual-target position $\boldsymbol{w}_j$ and $\boldsymbol{w}_{j+1}$ (Figure 4). The smoothness of the trajectory can be modified in two ways: (i) locally by adjusting the length of an additional handle placed at the same position, the length of which interpolates Δt_j between Δt_{min} and Δt_{max}, or (ii) globally by varying the Δt_{min} and Δt_{max} parameters.

In the case of a $\Sigma\Lambda^\star$ trajectory the interaction method is nearly identical, the only difference being that the user can only vary the ϕ_j parameters (instead of ϕ_{1j} and ϕ_{2j} for the $\Gamma\Lambda^\star$ case) with a single handle placed around each virtual target $\boldsymbol{w}_j$.

The velocity information generated by the gestural model (Figure 6, top) adds a layer of information to the curve that can be used to render realistic brush patterns and also to generate brush stroke animations. As an example, we have implemented a simple painterly rendering technique in which a textured brush is swept along the generated trajectory. The size of the brush is determined with an inverse function of the velocity at each point on the curve. We then add a random drip effect where the velocity is under a certain threshold (Figure 6, bottom); this mimics a feature that can often be seen in instances of tags made with an ink marker or spray paint. Furthermore, the distances between consecutive positions along the generated trajectories are proportional to the velocity generated by the model. As a result, by simply incrementally sweeping the brush along the trajectory at a fixed time step, we can achieve a realistic animation of the trace's evolution over time.

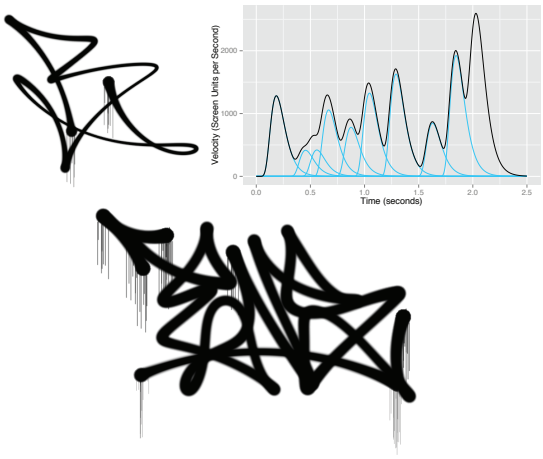

Figure 6: *Top: A stylised letter B generated with the $\Sigma\Lambda^*$ model and the corresponding velocity plot; the velocity of each strokes is plotted in turquoise. Bottom: a tag interactively specified with the $\Gamma\Lambda^*$ model. Both tags incorporate a speed-related drip effect.*

6. Gesture Reconstruction

The previously demonstrated method allows us to easily define the shape and velocity of trajectories that result in graffiti-like outputs. We now describe a method that allows the extraction of the same $\Gamma\Lambda^\star$ parameters from an arbitrary sequence of planar points. The input may result from a variety of sources, ranging from input devices as a mouse, trackpad or tablet, to vector art or contours extracted from bitmap images or video. Although the input may provide time/velocity information, we purposely choose not to take this into account in order to seamlessly treat online and offline data with the same method. In practice, the system we develop infers a plausible gesture and its velocity from a (possibly) static input trace.

Like the system described by Li, Parizeau and Plamondon [LPP98], our method relies on the segmentation of the input contour in correspondence with a series of *dominant* points (Figure 7, left). The segmentation process will produce m points and $m-1$ sections of the contour, each delimited by an adjacent pair of dominant points. The resulting trajectory will also be made of $(m-1)$ $\Gamma\Lambda^\star$ strokes and a corresponding action plan made of m points (Figure 7, right). The fitting algorithm can be summarised with the following steps:

1. Find m dominant points and $m-1$ contour sections.
2. For each jth section compute the corresponding ϕ_{1j} and ϕ_{2j} parameters.

3. For each jth section compute the corresponding Δt_j.
4. Set an initial action plan with m virtual target positions corresponding to each dominant point.
5. Compute the $\Gamma\Lambda^*$ trajectory.
6. Move the virtual target positions to minimise the error between the generated trajectory and the input contour.
7. Repeat 5 and 6 until convergence or until a maximum number of iterations is reached.

Here convergence is measured as the max error allowed (below some a priori set threshold). The algorithm relies on the same simplifying assumptions and parameters as the interactive trajectory specification case (Section 4), so we keep μ and σ constant across strokes and define a Δt range between Δt_{min} and Δt_{max}.

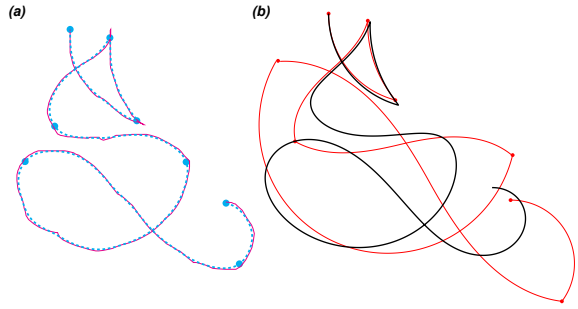

Figure 7: *Fitting a $\Gamma\Lambda\star$ trajectory to a sketched letter S.* ***(a)*** *A shaky input trace (magenta), the smoothed input trace (dashed turquoise) and the corresponding dominant points (turquoise circles).* ***(b)*** *The reconstructed trajectory and the corresponding action plan (red).*

6.1. Segmentation

Our system takes as an input a contour C uniformly sampled at n steps of constant length $\Delta s = 1$. we denote each ith sampled point in C with $\boldsymbol{p}(i)$. As a preprocess, we perform a weighted averaging on the points of C; this is done in order to remove discontinuities and noise that may be caused for example by latency in the input. We create an initial set of dominant points by finding the salient extrema of curvature along the contour.

Finding significant curvature extrema (which can be counted as convex and concave features) is an active area of research, as relying on discrete curvature measurements remains challenging. This is an aspect of our work we plan to refine in the future.

We currently use a method similar to the one described by De Winter & Wagemans [DWW08]: first we measure the turning angle $\theta(i)$ at each position $\boldsymbol{p}(i)$ and then find the local maxima for $-cos(\theta(i))$. This, plus the initial and final points along the contour gives us a first estimate of the salient points. We then select a subset of these points by measuring the angle between each point and its two adjacent neighbours and keeping the salient points with an angle greater than an experimentally selected threshold (25°).

Successively, we compute a sum $\hat{\theta}$ of the discrete turning angles between each pair of salient points. If $\hat{\theta} > 180^\circ$, we add $\lfloor \hat{\theta}/180 \rfloor$ dominant points equally spaced along the contour and between the two salient points under examination.

This process gives us m dominant point indices $S = (z_j, j = 1, ..., m)$ and subdivides C in $m-1$ sections, each defined between $\boldsymbol{p}(z_j)$ and $\boldsymbol{p}(z_{j+1})$. Note that the input contour is not segmented in correspondence with inflection points. These will be identified in the following step, and captured by means of the Euler spiral stroke primitive.

6.2. Estimating ϕ_{1j} and ϕ_{2j}

For each jth section defined between $\boldsymbol{p}(z_j)$ and $\boldsymbol{p}(z_{j+1})$, we estimate the corresponding values for ϕ_{1j} and ϕ_{2j}. This is done by fitting either one or two circular arcs to the section, depending on the presence of an inflection along its contour. If an inflection is present we fit *two* circular arcs: one arc is defined between $\boldsymbol{p}(z_j)$ and the inflection; the second arc is defined between the inflection and $\boldsymbol{p}(z_{j+1})$. If there is no inflection, we fit *one* circular arc between $\boldsymbol{p}(z_j)$ and $\boldsymbol{p}(z_{j+1})$.

Now, let $\boldsymbol{p}$ and $\boldsymbol{q}$ be two points along the contour, we fit a circular arc to the corresponding contour section by computing the signed area A_Z of the section and finding a corresponding circular segment with chord length $c = |q - p|$ and area $A_s = |A_Z|$ (Figure 8). Given that the chord length of a circle can be computed with $c = 2r \times sin(\frac{1}{2}\theta)$ and the circular segment area with $A_s = \frac{1}{2}r^2(\theta - sin(\theta))$, the internal angle θ is then given by numerically solving the following equation for θ (*e.g.* using a Newton-Raphson scheme):

$$\frac{1}{2}\left(\frac{c}{2\sin\frac{\theta}{2}}\right)^2(\theta - \sin\theta) - A_s = 0 \quad . \tag{19}$$

If an inflection is present, we use eq. 19 to compute two internal angles (θ_1, θ_2) and then let $\phi_{1j} = -\frac{\theta_1}{2}$ and $\phi_{2j} = \frac{\theta_2}{2}$. With no inflection the section can be described by an Euler spiral approximation of a circular-arc stroke primitive and we find the internal angle θ and then set $\phi_{1j} = -\frac{\theta}{2}$ and $\phi_{2j} = -\phi_{1j}$.

6.3. Estimating Time Overlaps and Contour Sharpness

As previously stated, using a greater time overlap, *i.e* smaller values of Δt_j, results in a smoother trajectory. With a large enough value of Δt_j the trajectory will form a sharp corner in correspondence with the virtual target $\boldsymbol{w}_j$. Based on this notion, it is possible to infer a plausible value for each Δt_j by examining the sharpness of the input curve in a region Z around each dominant point $\boldsymbol{p}(z_j)$. The region we examine is defined by a support length l_{sup} and is delimited between the points $\boldsymbol{a} = \boldsymbol{p}(z_j - l_{sup})$ and $\boldsymbol{b} = \boldsymbol{p}(z_j - l_{sup})$.

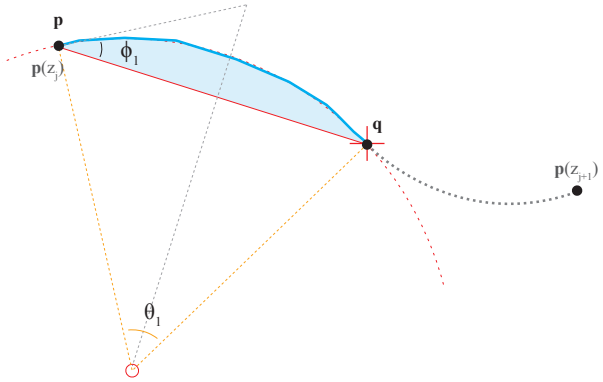

Figure 8: *Computing ϕ_1 by fitting a circular arc between the first point of the contour section $\boldsymbol{p}(z_j)$ and the inflection at $\boldsymbol{q}$. ϕ_2 will be computed similarly between the inflection (red cross) and $\boldsymbol{p}(z_{j+1})$.*

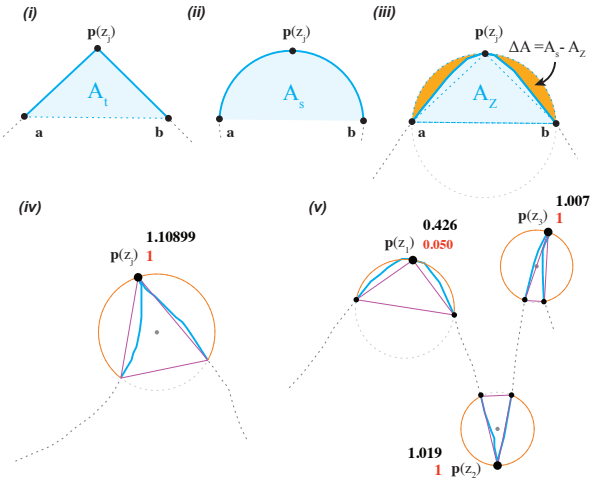

Figure 9: *Computing the sharpness of a contour section. **(i)** Maximally triangular case, the contour section Z is a triangle with area A_t. **(ii)** Maximally circular case, the contour section Z is a circular segment with area A_s. **(iii)** Sharpness estimation on a contour section with area A_Z. A sharper corner at $\boldsymbol{p}(z_j)$ will result in an increase of the area $\Delta A = A_s - A_Z$ (in orange) and in a consequently higher sharpness value. **(iv)** A case in which the area of the contour section is less than the corresponding triangle area. The value of $S(Z)$ will be > 1 (in black) and will be clamped to 1 (in red) with $min\left(S(Z)^{3.5},1\right)$ (eq. 21). **(v)** Example of various values of $S(Z)$ (in black) computed for the regions around 3 dominant points. In red, the corresponding value of $min\left(S(Z)^{3.5},1\right)$ (eq. 21).*

To estimate a notion of sharpness for the contour section Z we compute its area A_Z and compare its shape with a *maximally triangular* and *maximally circular* case. In the maximally triangular case (Figure 9.i), Z assumes the shape of a triangle with area A_t and vertices $(\boldsymbol{a},\boldsymbol{p}(z_j),\boldsymbol{b})$; the sides of the triangle form a sharp corner at $\boldsymbol{p}(z_j)$. In the maximally circular case (Figure 9.ii), Z assumes the shape of a circular-section with area A_s, chord $\boldsymbol{ab}$ and circumscribing the vertices $(\boldsymbol{a},\boldsymbol{p}(z_j),\boldsymbol{b})$. We then estimate the sharpness of Z with:

$$S(Z) = \begin{cases} (A_s - A_Z)/(A_s - A_t), & if A_s > \{A_Z, A_t\} \\ 0, & otherwise \ . \end{cases} \quad (20)$$

We then can estimate each Δt_j with:

$$\Delta t_j = \Delta t_{min} + (\Delta t_{max} - \Delta t_{min}) \min\left(S(Z)^a, 1\right) \ , \quad (21)$$

where the exponent a is used to bias $S(Z)$ towards smoother (lower) values with $a > 1$. During our experiments we have found that values of $a = 3.5$ give the best results (Figure 9.v).

We have experimented with various measures to determine the sharpness of a contour region, including estimating curvature and compactness [Zus70]. So far, the above method (using $S(Z)$) has proven experimentally to be the most robust to noise and produces the best results for our use case.

6.4. Iterative Fitting

The positions generated by the segmentation step also provide a first estimate for an action plan where each virtual target position is given by $\boldsymbol{w}_j = \boldsymbol{p}(z_j)$. Due to the smoothing effect given by the stroke time-overlaps, the initial action plan is likely to generate a trajectory that has a smaller scale with respect to the input contour [VKB05]. We overcome this problem with an iterative method that perturbs each virtual target towards a position that will seek to minimise the error between the generated trajectory and the input contour.

At each iteration, a sequence of critical points $(\boldsymbol{c}_j, 0 \leq j < m)$ along the generated trajectory is computed; these are the initial and final loci together with the positions where velocity profiles of each pair of consecutive strokes intersect (*i.e.* where $v_j(t) = v_{j+1}(t)$). These positions can be computed analytically (assuming that $\mu_j = \mu_{j+1}$ and $\sigma_j = \sigma_{j+1}$), but in practice, it is convenient to compute the intersections numerically during the trajectory integration process (Eq. 5). We move each critical point $\boldsymbol{c}_j$ towards the corresponding target point along the input contour $\boldsymbol{p}(z_j)$ by offsetting each virtual target $\boldsymbol{w}_j$ with:

$$\boldsymbol{w}_j = \boldsymbol{w}_j + \boldsymbol{p}(z_j) - \boldsymbol{c}_j \, , \quad (22)$$

The iteration continues until the Mean Square Error (MSE) of the distances between every pair $\boldsymbol{p}(z_j)$ and $\boldsymbol{c}_j$ is less than an experimentally set threshold or until a maximum number of iterations is reached (Figure 10). We have

empirically tested that offsetting a virtual target by a vector $\boldsymbol{d}$ will cause the motion of the corresponding critical point $\boldsymbol{c}_j$ by a vector $\boldsymbol{d}'_j$ where $|\boldsymbol{d}'_j| \leq |\boldsymbol{d}|$ and $\boldsymbol{d}'_j \cdot \boldsymbol{d} > 0$, as long as $|\boldsymbol{d}| > 1$. The distance between $\boldsymbol{p}(z_j)$ and $\boldsymbol{c}_j$ will rapidly decrease with each iteration and eventually oscillate around a value ≈ 1 screen units, which we choose as a threshold value. In practice this method gives satisfactory results after a small number of iterations, and for the sake of interactive performance we have chosen to limit the number of iterations to 10

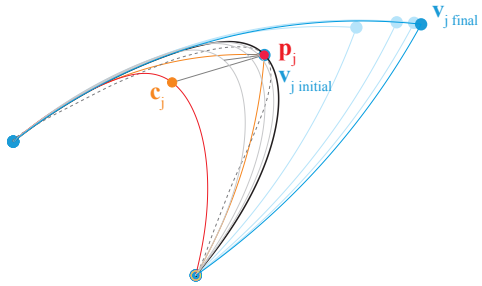

Figure 10: *Iterative fitting: The initial action plan (orange) generates a trajectory (red) that is scaled with respect to the original (dashed grey). Through successive iterations the initial virtual target position moves until the corresponding point along the trajectory (orange circle) converges towards the input segmentation point (red circle).*

6.5. $\Sigma\Lambda^\star$ Reconstruction

In order to fit $\Sigma\Lambda^\star$ parameters to an input contour, we use a process that is almost identical to the one we have described for $\Gamma\Lambda^\star$ case. The principal difference is that inflections along the input contour are computed during the segmentation step (§6.1) and are considered as additional dominant points.

6.6. Applications

The fitting technique runs at interactive rates, and allows the user to generate smooth graffiti-like curves with a simple gesture made with a mouse, a trackpad or a tablet. The resulting trajectory is generated immediately as soon as the input gesture is finished. A shaky and slowly made input trace will produce a trajectory that evokes the rapid motion of an expert hand (Figure 7). Once the trajectory has been fitted, the user can easily adjust the model parameters and virtual target positions (§5); to the best of our knowledge, this is a unique property of our system when compared to other existing methods aimed at curve beautification and stylisation.

7. Results

Our system is implemented in C++ and uses OpenGL for hardware accelerated rendering. The system allows us to interactively specify $\Gamma\Lambda$ and $\Sigma\Lambda$ trajectories, modify the parameters in real time and fit trajectories to input contours. The input can be defined with a gesture made with a mouse or tablet, or by loading contours from different types of files. The interactive design procedure makes it easy for a user to specify convincing images with a stylistic signature similar to graffiti. In this sense it is a stylistically aware design system, and we regard it as a potentially useful tool in the graphic design domain. In addition, a more expert user is able to take advantage of this kind of system to rapidly explore creative possibilities in the graffiti art style-space.

Figure 11: *Variations over a tag taken from the Graffiti Analysis database. In red, the original traces. Below, a variety of reconstructions with different parameter variations and dripping effect.*

7.1. Tag Reconstruction

In an additional experiment we reconstruct the trajectory of existing tags with the Gamma Lognormal model and then generate a variety of exemplars by perturbing parameters (Figure 11). We take the source data for the tags from the Graffiti Analysis database (http://000000book.com) which contains the motion data for thousands of tags stored

in a format named *GML* (Graffiti Markup Language) which is essentially an XML file storing a series of positions coupled with time stamps.

The Kinematic Theory allows us to generate multiple specimens from a single example with a variability that is similar to what we would expect to see in different instances of writing by the same artist [DP09]. We vary the parameters of the model within an empirically chosen threshold that will not disrupt the readability and visual quality of the tag. We create modified versions of the original reconstruction with the following variations: (i) we offset each virtual target position in the action plan by a random amount; (ii) we scale the action plan horizontally and vertically by a random amount; (iii) we randomly scale the Δt parameter of each stroke; (iv) we scale the angular deviation parameters ϕ_1 and ϕ_2 by a random amount.

As previously mentioned, we purposely ignore the timing information during the fitting process; we do so with the future purpose of applying the same method to traces extracted from a variety of (bitmap) images of graffiti. We envision a potential application of this kind of system in the domain of Procedural Content Generation in Games, for the rapid generation of a diversity of tags that can be then utilised for texturing graffiti in a virtual urban environment.

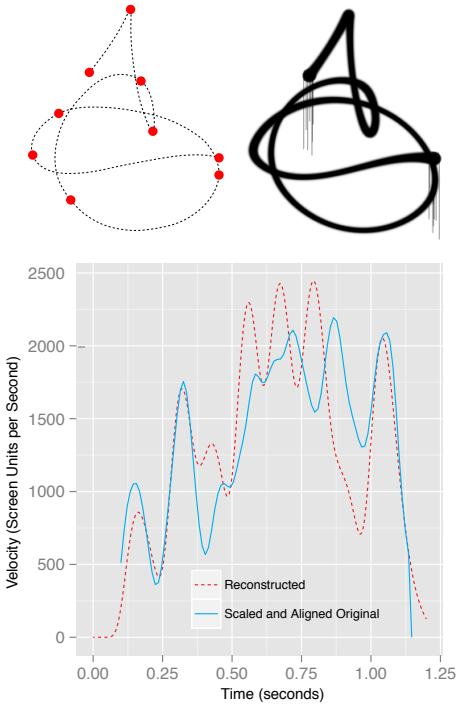

Figure 12: *Fitting a $\Gamma\Lambda^\star$ trajectory. Top Left: the input contour and corresponding segmentation points. Top Right: the generated trajectory. Bottom: The velocity of the generated trajectory (red) overlapped with a manually scaled and aligned version of the input velocity signal (turquoise) .*

8. Conclusion

In this communication we have described a system that is capable of generating traces that are similar, both kinetically and aesthetically, to the ones that are typically seen in well executed graffiti tags. By using a physically inspired model for curve generation, we gain an additional layer of information in the velocity domain. This is potentially useful for (i) the implementation of realistic brush patterns, (ii) generating realistic stroke animations, (iii) the smooth control of a robotic drawing effector as a pen plotter or robot arm (or even a humanoid character in a virtual world).

Kinematic Theory based models offer a concise and meaningful representation of the complex trajectory that can be seen in handwriting and drawing gestures. Concise, because a complex trajectory is described with a reduced number of parameters (stroke primitives). Meaningful, because the stroke primitives and their parameters reflect biologically plausible units of motor action. We plan to apply this type of representation in the future not only as an interactive design tool, but also for the procedural generation of realistic tags and, eventually, also more complex forms of graffiti art.

We observe that when the input to the system has been produced with a rapid gesture (as in the case of a gesture executed by an expert artist), the velocity signal reconstructed by the system is not an identical but a *plausible* reconstruction of the original (Figure 12). In future developments of this work, we plan to methodically compare the reconstructed velocity signals with the original ones, and evaluate the quality of the reconstruction.

Acknowledgements

We would like to thank Prashant Aparajeya for the many useful comments. This work has been partly supported by UK's EPSRC (grant EP/L015846/1) Centre for Doctoral Training in Intelligent Games and Game Intelligence (IGGI).

References

[AWI*09] ALMERAJ Z., WYVILL B., ISENBERG T., GOOCH A. A., GUY R.: Automatically mimicking unique hand-drawn pencil lines. *Computers & Graphics 33*, 4 (2009), 496–508. 3

[BAS04] BEZINE H., ALIMI A. M., SHERKAT N.: Generation and analysis of handwriting script with the beta-elliptic model. *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR 8*, 2 (2004), 515–520. 3

[BF13] BERTOLAZZI E., FREGO M.: Fast and accurate G^1 fitting of clothoid curves. *arXiv preprint arXiv:1305.6644* (2013). 6

[BGM93] BULLOCK D., GROSSBERG S., MANNES C.: A neural network model for cursive script production. *Biological Cybernetics 70*, 1 (1993), 15–28. 3

[CK14] CONNOR D., KRIVODONOVA L.: Interpolation of two-dimensional curves with Euler spirals. *J. Comput. Appl. Math. 261* (May 2014), 320–332. 6

[DP09] DJIOUA M., PLAMONDON R.: Studying the variability of handwriting patterns using the Kinematic Theory. *Human Movement Science 28*, 5 (2009), 588–601. 11

[DWW08] DE WINTER J., WAGEMANS J.: Perceptual saliency of points along the contour of everyday objects: A large-scale study. *Perception & Psychophysics 70*, 1 (2008), 50–64. 8

[eye09] Eye writer, 2009. URL: http://eyewriter.org. 3

[FG07] FREEDBERG D., GALLESE V.: Motion, emotion and empathy in esthetic experience. *Trends in cognitive sciences 11*, 5 (2007), 197–203. 2

[FH85] FLASH T., HOGAN N.: The coordination of arm movements: An experimentally confirmed mathematical model. *The journal of Neuroscience 5*, 7 (1985), 1688–1703. 3

[GG10] GILBRETH F. B., GILBRETH L. M.: *Applied Motion Study: A Collection of Papers On the Efficient Method to Industrial Preparedness*. Nabu Press, 2010. 2

[HEWF13] HAVEMANN S., EDELSBRUNNER J., WAGNER P., FELLNER D.: Curvature-controlled curve editing using piecewise clothoid curves. *Computers & Graphics 37*, 6 (2013), 764–773. 3

[HOCS02] HERTZMANN A., OLIVER N., CURLESS B., SEITZ S. M.: Curve analogies. In *Rendering Techniques* (2002), pp. 233–246. 3

[Hol81] HOLLERBACH J. M.: An oscillation theory of handwriting. *Biological Cybernetics 39*, 2 (1981), 139–156. 3

[KFP03] KIMIA B., FRANKEL I., POPESCU A.: Euler spiral for shape completion. *International journal of computer vision 54* (2003), 159–182. 6

[KHVG86] KAO H. S., HOOSAIN R., VAN GALEN G.: *Graphonomics: Contemporary research in handwriting*. Elsevier, 1986. 3

[Kim14] KIMVALL J.: *The G-word*. Dokument Press, Stockholm, 2014. 2

[Knu79] KNUTH D.: Mathematical typography. *Bulletin of the American Mathematical Society 1*, 2 (1979), 337–372. 4

[KY12] KANNO S., YAMAGUCHI T.: Sensless drawing bot. kanno.so/senseless-drawing-bot/, 2012. 3

[LARV03] LONGCAMP M., ANTON J. L., ROTH M., VELAY J. L.: Visual presentation of single letters activates a premotor area involved in writing. *NeuroImage 19*, 4 (2003), 1492–1500. 2

[Leh04] LEHNI J.: Hektor. http://hektor.ch/, 2004. 3

[Lev08] LEVIEN R.: The Euler spiral: A mathematical history. *Opera* (2008), 1–14. 5, 6

[Lev09] LEVIEN R. L.: *From Spiral to Spline: Optimal Techniques in Interactive Curve Design*. PhD thesis, EECS Department, University of California, Berkeley, Dec 2009. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-162.html. 6

[LPP98] LI X., PARIZEAU M., PLAMONDON R.: Segmentation and reconstruction of on-line handwritten scripts. *Pattern recognition 31*, 6 (1998), 675–684. 3, 7

[LTV83] LACQUANITI F., TERZUOLO C., VIVIANI P.: The law relating the kinematic and figural aspects of drawing movements. *Acta psychologica 54*, 1 (1983), 115–130. 2

[LYFD12] LU J., YU F., FINKELSTEIN A., DIVERDI S.: HelpingHand: Example-based stroke stylization. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 46. 3

[MMI82] MORASSO P., MUSSA IVALDI F.: Trajectory formation and handwriting: A computational model. *Biological cybernetics 45*, 2 (1982), 131–142. 3

[MS09] MCCRAE J., SINGH K.: Sketching piecewise clothoid curves. *Computers and Graphics (Pergamon) 33*, 4 (2009), 452–461. 3, 6

[PD06] PLAMONDON R., DJIOUA M.: A multi-level representation paradigm for handwriting stroke generation. *Human Movement Science 25*, 4 (2006), 586–607. 4

[PDO09] PLAMONDON R., DJIOUA M., O'REILLY C.: Recent Developments in the Study of Rapid Human Movements with the Kinematic Theory. *Traitement Du Signal 26* (2009), 377–394. 4

[PFW03] PLAMONDON R., FENG C., WOCH A.: A kinematic theory of rapid human movement. Part IV: A formal mathematical proof and new insights. *Biological Cybernetics 89*, 2 (2003), 126–138. 4

[Pla95] PLAMONDON R.: A kinematic theory of rapid human movements. Part I: Movement representation and generation. *Biological cybernetics 72*, 4 (1995), 295–307. 2, 3

[POG*14] PLAMONDON R., O'REILLY C., GALBALLY J., ALMAKSOUR A., ANQUETIL É.: Recent developments in the study of rapid human movements with the kinematic theory: Applications to handwriting and signature synthesis. *Pattern Recognition Letters 35* (2014), 225–235. 7

[PORD13] PLAMONDON R., O'REILLY C., REMI C., DUVAL T.: The lognormal handwriter: Learning, performing and declining. *Frontiers in Psychology 4*, 945 (2013). 4

[Rot04] ROTH E.: Graffiti analysis, 2004. URL: http://http://www.graffitianalysis.com/about/. 3

[SG97] STOWERS G. C., GOLDMAN P.: Graffiti art: An essay concerning the recognition of some forms of graffiti as art. *Unpublished essay, Fall* (1997). URL: http://www.graffiti.org/faq/stowers.html. 2

[TS93] TEULINGS H.-L., SCHOMAKER L. R.: Invariant properties between stroke features in handwriting. *Acta psychologica 82*, 1 (1993), 69–88. 3

[TSB11] THIEL Y., SINGH K., BALAKRISHNAN R.: 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* (2011), pp. 383–392. 3

[TT85] THOMASSEN A., TEULINGS H.-L.: Time, size and shape in handwriting: Exploring spatio-temporal relationships at different levels. In *Time, Mind, and Behavior*, Michon J., Jackson J., (Eds.). Springer Berlin Heidelberg, 1985, pp. 253–263. 7

[VC85] VIVIANI P., CENZATO M.: Segmentation and coupling in complex movements. *Journal of experimental psychology: Human perception and performance 11*, 6 (1985), 828. 2

[VKB05] VARGA T., KILCHHOFER D., BUNKE H.: Template-based Synthetic Handwriting Generation for the Training of Recognition Systems. In *Proc. of 12th Conf. of the International Graphonomics Society* (2005), pp. 206–211. 4, 9

[VM83] VIVIANI P., MCCOLLUM G.: The relation between linear extent and velocity in drawing movements. *Neuroscience 10*, 1 (1983), 211–218. 2, 7

[VS92] VIVIANI P., STUCCHI N.: Biological movements look uniform: Evidence of motor-perceptual interactions. *Journal of Experimental Psychology: Human Perception and Performance 18*, 3 (1992), 603–623. 2

[WM08] WALTON D., MEEK D.: An improved Euler spiral algorithm for shape completion. In *Canadian Conference on Computer and Robot Vision* (Windsor, Ontario, May 2008), pp. 237–244. 6

[WM09] WALTON D., MEEK D.: G1 interpolation with a single Cornu spiral segment. *Journal of Computational and Applied Mathematics 223*, 1 (2009), 86–96. 6

[XHLW14] XIE J., HERTZMANN A., LI W., WINNEMÖLLER H.: Portraitsketch: Face sketching assistance for novices. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (2014), pp. 407–417. 3

[Zit13] ZITNICK C. L.: Handwriting beautification using token means. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 53. 3

[Zus70] ZUSNE L.: *Visual perception of form*. Academic Press New York, 1970. 9