Calligraphic Stylisation Learning with a Physiologically Plausible Model of Movement and Recurrent Neural Networks

Daniel Berio
Goldsmiths, University of London
d.berio@gold.ac.uk

Memo Akten
Goldsmiths, University of London
m.akten@gold.ac.uk

Frederic Fol Leymarie
Goldsmiths, University of London
ffl@gold.ac.uk

Mick Grierson
Goldsmiths, University of London
m.grierson@gold.ac.uk

Réjean Plamondon
Ecole Polytechnique de Montréal
rejean.plamondon@polymtl.ca

ABSTRACT

We propose a computational framework to *learn* stylisation patterns from example drawings or writings, and then generate new trajectories that possess similar stylistic qualities. We particularly focus on the generation and stylisation of trajectories that are similar to the ones that can be seen in calligraphy and graffiti art. Our system is able to extract and learn dynamic and visual qualities from a small number of user defined examples which can be recorded with a digitiser device, such as a tablet, mouse or motion capture sensors. Our system is then able to transform new user drawn traces to be kinematically and stylistically similar to the training examples. We implement the system using a Recurrent Mixture Density Network (RMDN) combined with a representation given by the parameters of the Sigma Lognormal model, a physiologically plausible model of movement that has been shown to closely reproduce the velocity and trace of human handwriting gestures.

CCS CONCEPTS

•**Computing methodologies** →**Neural networks; Procedural animation;** •**Human-centered computing** →*Gestural input;*

KEYWORDS

Human hand-writing movement modeling; procedural calligraphic and graffiti production; graffiti tags; curve and path dynamics; Recurrent Neural Network, RNN; Long Short-Term Memory architecture, LSTM; Mixture Density Network, MDN; Recurrent Mixture Density Network, RMDN; graphonomics; one-shot learning.

1 INTRODUCTION

When writing or drawing, the pen or brush leaves a static mark which is a record of the movement underlying its production. Each such static trace inherits qualities directly due to the generating

movement, and is further unique to each individual. There is cumulating cognitive and neuroscientific evidence to suggest that, as observers, humans implicitly (at a neural level) reconstruct a movement when observing such traces on a canvas [34, 43]. Furthermore, while a given letter may be written with different topological structures — consider for example a lower case "a" written as a cursive "α" — and different ordering of generative gestures, the same topology and gesture ordering may themselves be executed with a possibly infinite number of variations. Such variations are principally determined by the movement dynamics used in the production of a letter. In our study we will consider a particular model of movement when writing as a powerful descriptor of various stylisations when applied to a common topological structure of a letter form.
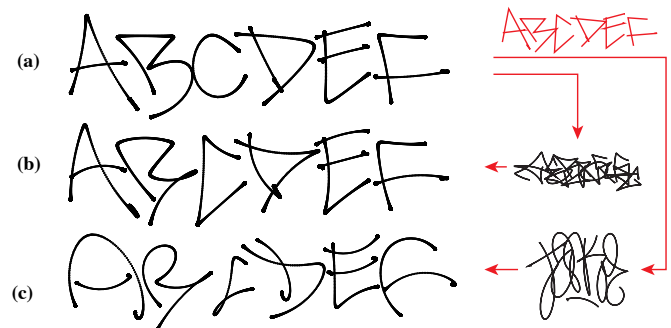

Figure 1: Stylisation of a user drawn trace and one-shot learning. (a) An input trace drawn by a user (left) and the extracted virtual targets (right, in red). (b,c) We train two DPP models, each on a single example (right). Each model predicts new dynamic parameters for the user drawn trace (a) to generate the trajectory on the left.

We base our work on the Sigma Lognormal model which is recognised as a physiologically plausible representation of the kinematics of hand movements [37]. This model facilitates the implementation of a bi-level representation of a written trace made of a *structural* and a *dynamic* component. The *structural* component defines the overall spatial layout of the handwriting trace through a coarse ordered sequence of targets. Such targets define the aiming position for a sequence of consecutive goal directed sub-movements (*aka* movement primitives or strokes). The *dynamic* component determines the velocity, acceleration, timing and curvature of each sub-movement, and produces a smooth trajectory that follows the

spatial layout of the targets. By exploiting this bi-level representation, we train a type of Recurrent Neural Network (RNN) that learns correlations between the structure of a trace and the corresponding parameters that control its fine dynamic evolution (Fig. 1).

To learn these correlations, we use *Recurrent Mixture Density Networks (RMDNs)* with *Long Short-Term Memory (LSTM)* architecture. Recent developments have shown that such types of RNNs are capable of modelling complex sequential (or time-ordered) data such as text [42], images [18], dance [8] and handwriting [17]. While most existing 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 [15]. As a result, for this particular study, we seek a mapping that 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.

In this paper, we demonstrate our approach on digitised traces, which are recorded by a user with a pen-tablet or downloaded from a large online graffiti motion database (GML, [25]). We first transform the input into an intermediate representation given by the parameters of the Sigma Lognormal model [37]. Doing so requires a preprocessing step which results in a representation that is more concise (*i.e.* with low cardinality) and meaningful, such that every representative locus is now a high level segment of the trajectory when compared to an initial sequence of positions. Such a representation is also *resolution independent*, and can easily be manipulated prior and after training. For example, it is possible to introduce random perturbations at the model parameter level, which results in 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 input data-sets — as small as a single training example, as we will demonstrate (aka one-shot learning).

In the following, after summarising the background (§2), we briefly describe the Sigma Lognormal model, present the data preprocessing step and the implemented RMDN model (§3); then we propose various applications of the system, including learning handwriting representations from small datasets and mixing styles (§4).

2 BACKGROUND

Our study is grounded on a number of notions and principles that have been observed in the general study of human movement as well as in *graphonomics*, the handwriting synthesis and analysis scientific field [24]. The speed profile of aiming movements is typically characterised by a "bell shape" that is variably skewed depending on the speed [29]. Complex movements can be described by the superimposition of a discrete number of "ballistic" units of motion, which in turn can each be represented by the classic bell shaped velocity profile and are often referred to as *strokes*. A number of methods synthesise handwriting through the temporal superimposition of strokes, the velocity profile of which is modelled with a

variety of functions including sinusoidals [40], Beta functions [6], and lognormals [35]. In our work we rely on a family of models known as the *Kinematic Theory of Rapid Human Movements*, mainly developed by R. Plamondon *et al.* in an extensive body of work since the 90's [35, 37]. They show that if we consider that a movement is the result of the parallel and hierarchical interaction of a large number of coupled linear systems, the impulse response of such a system to a centrally generated command asymptotically converges to a lognormal function. This assumption is attractive from a modelling perspective because it abstracts the high complexity of the neuromuscular system in charge of generating movements with a relatively simple mathematical model, which further provides state of the art reconstruction of human velocity data [35, 39].

A number of methods have used neural inspired approaches for the task of handwriting trajectory formation. With a preprocessing step similar to ours, Ltaief et al. [28] use a neural network to learn the mapping between the parameters of a handwriting model [6] and the corresponding trajectory. The network is then used instead of the model as a trajectory generator. Nair and Hinton [30] use a sequence of neural networks to learn the motion of two orthogonal mass spring systems from images of handwritten digits. The authors also propose that a motor representation can be exploited to generate realistically augmented training sets by introducing noise at the motor level. Plamondon and Privitera [38] 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. More recently, Graves [17] has demonstrated that the LSTM RNN combined with MDNs and trained on a large online dataset, are able to synthesise convincing handwriting sequences. His method provides the basis for a number of extensions [20, 45]. In our contribution, we design an RNN architecture inspired by the one studied by Graves together with a motor representation similar to the one of Plamondon and Privitera [38], and with an original goal of *trajectory stylisation*. While Graves works with a dense sequence of points sampled at fixed frequency, our method operates on a sparse and concise representation, which provides more flexibility and allows for meaningful manipulations that would be difficult to achieve otherwise. We demonstrate and discuss these in the remainder of the paper.

In the computer graphics domain, a number of data driven methods have been proposed for the task of vector path stylisation (e.g. [21, 26]). These method consider stylisation as a bi-level transfer problem: given two sketches, generate a third one with the structure or "content" of the first but with a style similar to the second. Our approach is also organised to function via a bi-level mapping, but focused on the motor level, such that we represent (i) "content" with a coarse structural description of a drawn trace, and (ii) "style" with different types of movements that follow the same structure. In the work reported here, we approach the problem with a data driven method, and learn such a bi-level mapping from user provided examples. In a complementary line of work [3, 4], we approach a similar problem from a procedural perspective, and use optimal control to define different stylisations of a given geometric structure.

3 METHOD

Our intermediate representation relies on the Sigma Lognormal ($\Sigma\Lambda$) model [35, 37], which generates trajectories that closely resemble the geometry [37] and kinematics [36, 39] of those made by an experienced human writer. The evolution of a pen-tip trajectory is then described with the spatio-temporal superimposition of a discrete number of ballistic strokes, where each consecutive stroke is aimed at an imaginary position denoted as a *virtual target*. We define our bi-level representation with (i) the structural component given by a *sequence of virtual targets*, and (ii) the dynamic component given by a subset of the remaining $\Sigma\Lambda$ parameters which define the fine evolution of the trajectory and which we refer to as *dynamic parameters* (Fig. 2).

The input to the system is given by a discrete and temporally ordered sequences of planar coordinates which are recorded with a digitizer device, such as a tablet or mouse. As a preliminary step, we preprocess this input data and reconstruct it in the form of $\Sigma\Lambda$ model parameters (§3.2). We then use this representation to train an RMDN model that learns to predicts the dynamic parameters corresponding to a given sequence of virtual targets (*Dynamic Parameter Prediction* or DPP, §3.4). In order to train on small datasets, we augment the preprocessed data by introducing artificial variability at the $\Sigma\Lambda$ parameter level. In the following sections we describe the steps that constitute our system and then demonstrate its use case with a number of examples (§4).
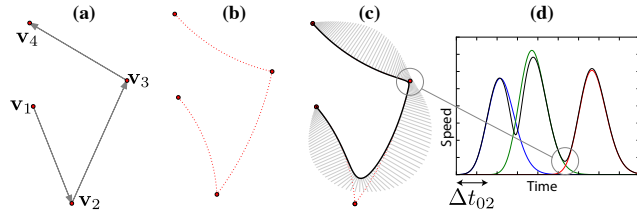
3.1 Sigma lognormal model

Figure 2: A sequence of virtual targets and the corresponding $\Sigma\Lambda$ trajectory. (a) Virtual targets and corresponding stroke aiming directions. (b) Corresponding circular arcs. (c) Possible trajectory generated over the given sequence of virtual targets. While the generated trajectory appears visually similar to an interpolating spline, it further describes a smooth and physiologically plausible *velocity* profile (d).

The $\Sigma\Lambda$ model explicitly describes the velocity profile of a stroke with a 3 parameter lognormal function:

$$\Lambda_i(t) = -\frac{1}{\sigma_i\sqrt{2\pi}(t - t_{0i})}\exp\left(\frac{(ln(t - t_{0i}) - \mu_i)^2}{2\sigma_i^2}\right), \quad (1)$$

where t_{0i} defines the activation time of a stroke and the parameters μ_i and σ_i determine the shape of the lognormal function. A detailed discussion of the biological interpretation of these parameters is presented by Plamondon [35]. *Assuming* that curved handwriting movements result from rotating the wrist, strokes are described by circular arcs:

$$\phi_i(t) = \theta_i + \theta_i\left[1 + \text{erf}\left(\frac{\ln(t - t_{0i}) - \mu_i}{\sigma_i\sqrt{2}}\right)\right], \quad (2)$$

where θ_i is the central angle of the arc for the ith stroke. A complete trajectory with m strokes is defined with an initial position $\boldsymbol{v}_0$ followed by a *sequence of virtual targets* $\{\boldsymbol{v}_i\}_{i=1}^{i=m}$, where a trajectory with m virtual targets will be characterised by $m - 1$ circular arc strokes. The trace can be then integrated with:

$$\xi(t) = \boldsymbol{v}_1 + \int_0^t d\tau\Lambda_i(\tau)\sum_{i=1}^{m-1}\Phi_i(\tau)\,(\boldsymbol{v}_{i+1} - \boldsymbol{v}_i), \quad (3)$$

where $\quad \Phi_i(t) = \begin{bmatrix} h(\theta_i)\cos\phi_i(t) & -h(\theta_i)\sin\phi_i(t) \\ h(\theta_i)\sin\phi_i(t) & -h(\theta_i)\cos\phi_i(t) \end{bmatrix},$ (4)

and $\quad h(\theta_i) = \begin{cases} \frac{2\theta_i}{2\sin\theta_i} & \text{if } |\sin\theta_i| > 0, \\ 1 & \text{otherwise,} \end{cases}$ (5)

which scales the extent of the stroke based on the ratio between the perimeter and the chord length of a circular arc.

Explicit asymmetry parameterisation. In order to facilitate the precise specification of timing and profile of each stroke, we explicitly define the structure of each lognormal profile through its duration T_i, a skewness parameter A_{c_i} and a time offset Δt_i with respect to the onset of the preceding stroke. The corresponding $\Sigma\Lambda$ parameters $\{t_{0i}, \mu_i, \sigma_i\}$ are then obtained as:

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

$$t_{0i} = t_{1i} - e^{\mu - 3\sigma}, \qquad t_{1i} = t_{1(i-1)} + d_{i-1}\Delta t_i, \qquad t_{1(0)} = 0, \quad (7)$$

where t_{1i} is the onset time of the lognormal stroke profile.

Dynamic parameters. In our method, we use the pairs $(\Delta t_i, \Theta_i)$ as the *dynamic parameters* that are mapped to a corresponding sequence of virtual targets. We keep the remaining parameters A_{c_i} and T_i (or accordingly μ_i, σ_i) fixed to predefined values (experimentally set to 0.1 and 0.3). While this is a simplifying assumption, these parameters mostly determine temporal properties of the writing motion, and thus do not produce a strong effect on the shape of the resulting trajectory.

3.2 Preprocessing

A number of methods have been developed by Plamondon *et al.* in order to reconstruct $\Sigma\Lambda$-model parameters from digitised pen input data [13, 31, 37]. These methods provide the ideal reconstruction of model parameters, given a high resolution digitised pen trace. While such methods are superior for handwriting analysis and biometric purposes, we opt for a less precise method [5] that is less sensitive to sampling quality and is aimed at generating virtual target sequences that remain perceptually similar to the original trace. We purposely choose to *ignore the original dynamics* of the input, and base the method on a geometric trace only. This is done in order to work with training sequences that are independent of sampling rate. Our method operates on an uniformly sampled input trace $\boldsymbol{p}[k]$, where the sampling distance depends on the maximum extent of the trace's bounding box. As an initial step (i) the trace is segmented in correspondence with perceptually salient *input key points*: i.e. loci of curvature extrema modulated by neighbouring contour segments [5]. The key points define the number of strokes in the $\Sigma\Lambda$ trajectory and provide an initial estimate of virtual target positions. We then (ii) fit a circular arc to each segment in order to estimate the θ_i parameters and (iii) estimate the Δt_{0i} parameters

by analysing the trace curvature in the region of each key point. Finally, (iv) we iteratively adjust the virtual target positions to minimise the error between the original trajectory and the one generated by the corresponding $\Sigma\Lambda$ parameters. In the following paragraphs we describe the steps in more detail.

(i) Estimating input key-points. To compute salient points along the input trace we rely on a method described by Fedman and Singh [12] and supported experimentally by De Winter *et al.* [10]. We then find local maxima in the signal:

$$s[k] = 1 - \cos(\theta[k]), \tag{8}$$

where $\theta[k]$ is the turning angle of the kth sample of the input, convolved with a Hanning window of size 40 to smooth noise in the input. The first and last sample indices of $s[k]$ plus its local maxima results in m key-point indices $\{\hat{z}_i\}$ with coordinates $\{\boldsymbol{p}[\hat{z}_i]\}$.

(ii) Estimating stroke curvature parameters. For each section of the input trace defined between two consecutive key-points, we estimate the corresponding stroke curvature parameter θ_i by first computing a least square fit of a circle to the trace segment (Fig. 3). We then compute the internal angle of the arc supported between the two key-points, which is equal to $2\theta_i$.
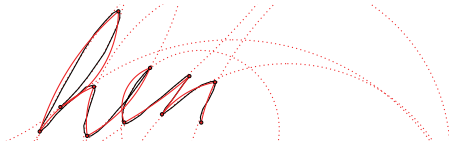

Figure 3: Fitting arcs (and associated circles as dotted lines).

(iii) Estimating stroke time-overlap parameters. This step is based on the observation that smaller values of Δt_{0i}, *i.e.* a greater time overlap between strokes, result in smoother trajectories. On the contrary, a sufficiently large value of Δt_{0i} will result in a sharp corner in proximity of the corresponding virtual target. We exploit this property to compute an estimate of the Δt_{0i} parameters by examining the sharpness of the input trace in the neighbourhood of each key-point. To do so we consider the previously computed signal $s[k]$ (8), in which we can observe that sharp corners correspond with sharp signal peaks, while smoother corners correspond with smooth peaks having a larger spread. By treating the signal as a probability density function, we can then employ a variant of Expectation Maximisation (EM) [11] to measure the shape of each peak with a mixture of parametric distributions (Fig. 4) — the reader is referred to recent work by Berio *et al.* [2] for additional details on the EM variant. We then examine the shape of each mixture component in order to get an estimate of the corresponding sharpness along the input trace. We further generate an estimate of sharpness λ_i (bounded in the $[0, 1]$ range) for each key point using a logarithmic function of the mixture parameters and weights. The corresponding Δt_{0i} parameters are then given by:

$$\Delta t_i = \Delta t_{min} + (\Delta t_{max} - \Delta t_{min})\lambda_i \ , \tag{9}$$

where Δt_{min} and Δt_{max} are user specified parameters that determine the range of the Δt_{0i} estimates. For the sharpness estimation function we currently use a mixture of Gaussians and empirically defined thresholds for the estimation λ_i. While this gives satisfactory results for our use-case, further studies are needed to limit
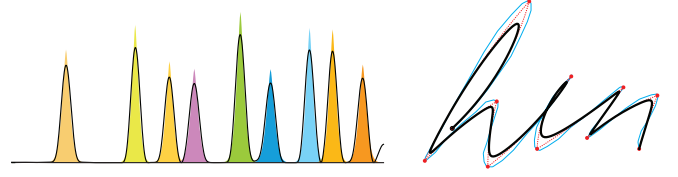

Figure 4: Sharpness estimation. Left, the GMM components (coloured) estimated from the turning angle signal (black) computed with Eq. (8). Right, the $\Sigma\Lambda$ trajectory generated before the final iterative adjustment step. Note that at this stage the virtual target positions correspond with the estimated input key-points.

the number of free parameters for this step. In future work we plan to test different types of distributions, such as generalised Gaussians, to learn the mapping between sharpness and mixture component parameters from synthetically samples generated with the $\Sigma\Lambda$ model, for which Δt_{0i}, and consequently λ_i, are known.

(iv) Iteratively estimating virtual target positions. The loci of the input key-points provide an initial estimate for a sequence of virtual targets, where each virtual target position is given by $\boldsymbol{v}_i = \boldsymbol{p}[\hat{z}_i]$. Due to the trajectory-smoothing effect produced by the time overlaps, the initial estimate will result in a generated trajectory that is likely to have a reduced scale with respect to the input we wish to reconstruct [44]. In order to produce a more accurate reconstruction we compute an estimate of *m output* key-points $\{\xi(z_i)\}$ in the generated trajectory, where $z_2, ..., z_m$ are the time occurrences at which the influence of one stroke exceeds the previous. These will also correspond with salient points along the trajectory (extrema of curvature) and can simply be computed by finding the time occurrence at which two consecutive lognormals intersect. Similarly to the input key-point case, $\xi(z_1)$ and $\xi(z_m)$ respectively denote the first and last points of the generated trajectory. We then iteratively adjust the virtual target positions in order
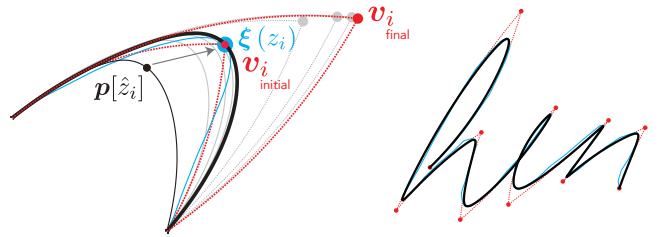

Figure 5: Final trajectory reconstruction step. Left, iterative adjustment of virtual target positions. Right, the final trajectory generated with the reconstructed dynamic parameters.

to move each generated key-point $\xi(z_i)$ towards the corresponding input key-point $\boldsymbol{p}[\hat{z}_i]$ with:

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \boldsymbol{p}[\hat{z}_i] - \xi(z_i), \tag{10}$$

The iteration continues until the Mean Square Error (MSE) of the distances between every pair $\boldsymbol{p}[\hat{z}_i]$ and $\xi(z_i)$ is less than an experimentally set threshold or until a maximum number of iterations is
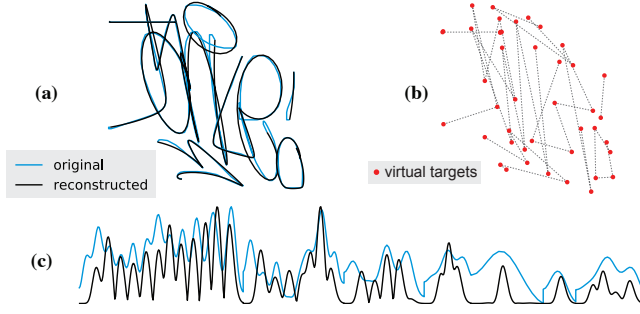
Figure 6: $\Sigma\Lambda$ parameter reconstruction. (a) Original (black) and reconstructed (turquoise) trajectories. (b) Reconstructed virtual targets. (c) Aligned and scaled speed profiles of the original and reconstructed trajectories.

reached (Fig. 5). This method usually converges to a good reconstruction of the input within few iterations (experimentally < 5). Interestingly, even though the dynamic information of the input is discarded, the reconstructed velocity profile is typically similar to the original in the number and outline of peaks (Fig. 6), which may be related to the extensively studied relationships existing between geometry and dynamics of movement trajectories [14].

3.3 Data augmentation

We can exploit the $\Sigma\Lambda$ parameterisation to generate many variations over a single trajectory, which are visually consistent with the original, and with a variability that is similar to the one that would be seen in multiple instances of handwriting made by the same writer (Fig. 7) [5, 13]. Given a dataset of n training samples, we randomly perturb the virtual target positions and dynamic parameters of each sample n_p times, which results in a new augmented dataset of size $n + n \times n_p$ where legibility and trajectory smoothness is maintained across samples. 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.

Figure 7: Data augmentation step.

3.4 Dynamic Parameter Prediction (DPP)

Given a sequence of virtual targets, we would like to predict the corresponding dynamic parameters. We train a model to learn the probability distribution of the dynamic parameters $\{\Delta t_{0i}, \theta_i\}$ for the ith stroke, conditioned on the virtual targets and dynamic parameters leading up to that stroke:

$$\Pr(\Delta t_{0i}, \theta_i \mid \Delta \boldsymbol{v}_{1:i}, u_{1:i}, \Delta t_{0(1:i-1)}, \theta_{1:i-1}), \qquad (11)$$

where $\Delta \boldsymbol{v}_i \in \mathbb{R}^2$ denotes the relative position displacement (between the ith virtual target and the next), $u_i \in \{0, 1\}$ denotes the pen-up state, and the subscript notation $a : b$ denotes all indices from a to b inclusive. Note that the distribution is conditioned

on the dynamic parameters of the *previous stroke* $(\Delta t_{0i-1}, \theta_{i-1})$. Conceptually this represents a writer that knows their next virtual target, is aware of the dynamic history of their movement so far, and wants to know what dynamic parameters to use.

We also considered an alternative model conditioned only on the virtual targets, independent of the previous dynamic parameters, i.e. $\Pr(\Delta t_{0i}, \theta_i \mid \Delta \boldsymbol{v}_{1:i}, u_{1:i})$, 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.

We implement our model using *Recurrent Mixture Density Networks (RMDN)* with the LSTM architecture. An MDN [7] models and predicts the parameters of a *Gaussian Mixture Model (GMM)*, i.e. a vector of means, covariances and mixture weights. An RMDN [41] is a combination of an RNN with an MDN, and outputs a *unique set of GMM parameters at each timestep*, allowing the probability distribution to change with time as the input sequence develops.
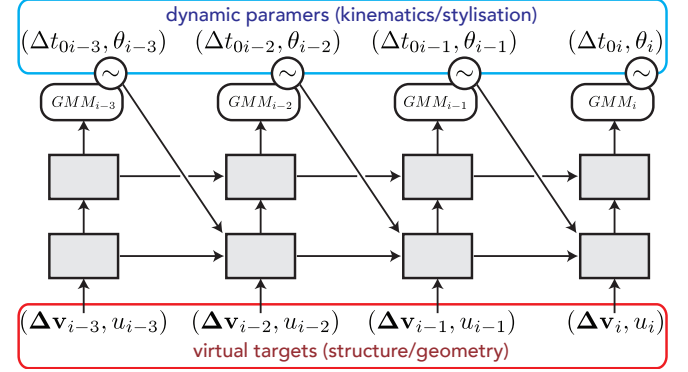

Figure 8: Network architecture with two recurrent hidden layers. At each timestep i the network outputs the parameters of a GMM which is sampled (denoted by $\sim$) and fed as input at the next timestep.

At each timestep i, the input to the model is given by a vector $\boldsymbol{x}_i \in \mathbb{R}^5$ corresponding to $[\Delta \boldsymbol{v}_i^\top, u_i, \Delta t_{0(i-1)}, \theta_{i-1}]$ where all elements (except for u_i) are normalised to zero mean and unit standard deviation. Our desired output is $\boldsymbol{z}_i \in \mathbb{R}^2$ which consists of the normalised dynamic parameters for the i'th stroke $\{\Delta t_{0i}, \theta_i\}$ (Fig. 8). Since $\boldsymbol{z}_i$ can be expressed as a *bivariate* GMM, we can use a network architecture with output dimensions of $6K$ where K represents the number of Gaussians. This results in an output vector $[\hat{\boldsymbol{\mu}}_i \in \mathbb{R}^{2K}, \hat{\boldsymbol{\sigma}}_i \in \mathbb{R}^{2K}, \hat{\boldsymbol{\rho}}_i \in \mathbb{R}^K, \hat{\boldsymbol{\pi}}_i \in \mathbb{R}^K]$ which we use to calculate the parameters of the GMM with (after Graves [17]):

$$\boldsymbol{\mu}_i^k = \hat{\boldsymbol{\mu}}_i^k : \text{means for } k\text{'th Gaussian}, \boldsymbol{\mu}_i^k \in \mathbb{R}^2,$$

$$\boldsymbol{\sigma}_i^k = \exp(\hat{\boldsymbol{\sigma}}_i^k) : \text{standard deviations for } k\text{'th Gaussian}, \boldsymbol{\sigma}_i^k \in \mathbb{R}^2,$$

$$\rho_i^k = \tanh(\hat{\rho}_i^k) : \text{correlations for } k\text{'th Gaussian}, \rho_i^k \in (-1, 1),$$

$$\pi_i^k = \text{softmax}(\hat{\pi}_i^k) : \text{mix weight for } k\text{'th Gaussian}, \sum_k^K \pi_i^k = 1.$$
$$(12)$$

We then formulate the probability of dynamic parameters $\boldsymbol{y}_i$ at timestep i, given input vector $\boldsymbol{x}_i$ as:

$$P_i(\boldsymbol{y}_i \mid \boldsymbol{x}_i) = \sum_{k}^{K} \pi_i^k \mathcal{N}(\boldsymbol{y}_i \mid \boldsymbol{\mu}_i^k, \boldsymbol{\sigma}_i^k, \rho_i^k). \qquad (13)$$

We efficiently compute the bivariate Gaussian without requiring matrix inversions by exploiting its closed form solution [17]. As our training objective, we use the *Maximum Likelihood*, minimising the *Negative Log Likelihood* (also known as *Hamiltonian* or *surprisal* [27]). Given a training set of input-target pairs $(\boldsymbol{x}, \hat{\boldsymbol{y}})$, we define the loss for a single training example with:

$$J_s = -\sum_{i}^{L} \ln P_i(\hat{\boldsymbol{y}} \mid \boldsymbol{x}), \qquad (14)$$

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) [42] 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 timesteps. 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 timesteps 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 [9] with the recommended hyperparameters. To prevent exploding gradients we clip these using the L2 norm as described by Pascanu et al. [32] and experimentally set the threshold.

We use LSTM networks [22] with input, output and forget gates [16], and dropout regularization [33]. We have employed both a grid and a random search [1] on various hyperparameters 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, each with size 400, 20 Gaussians, dropout keep probability of 90% and no peepholes.

4 EXPERIMENTS AND RESULTS

Given a sequence of virtual targets, the DPP model (§3.4) is used to produce different stylisations. The virtual targets can either be (i) directly defined by a user, (ii) reconstructed from an input path, or (iii) procedurally generated.

4.1 User defined virtual targets.

In this application of the system, a user can quickly sketch trajectories by interactively defining a sparse sequence of virtual targets. The system will then predict various smooth trajectories, the stylisation of which varies according to the dynamic parameters learned from a given training set. While the training procedure is performed offline, usually taking a few hours on a typical GPU, 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
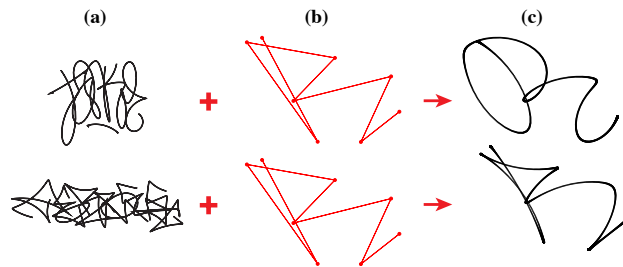

Figure 9: Dynamic parameters generated over user specified virtual targets (in red), using *two separate models*, each trained on a single example (with data augmentation x8000). Each row shows: (a) the training example, (b) the user provided virtual targets and (c) the trajectory predicted by the corresponding model. Training examples: top row is from the GML database, bottom row was drawn using a tablet.

dragging the virtual targets around with a mouse, and seeing the final smooth trajectory update instantaneously with desired styles.
One-shot learning. The training set can contain one or more examples of target styles. As mentioned previously, we can use data augmentation to train a DPP model on as few as a *single* initial training example, *aka* one-shot learning, and the model is able to consistently predict dynamic parameters in that style (Fig. 9).
Priming We also train a DPP model on a data-set that contains multiple styles. In this case we can control the stylisation by *priming* the model with the specific training example corresponding to the desired style (Fig. 10). Priming [17] is achieved by first feeding the model a training example with the desired style, before feeding it the virtual targets which we wish to make a prediction on. We observe that a shorter sequence length (e.g. 15) is sufficient for models trained on a single style example. However, for models trained on multiple styles, it is necessary to use a longer sequence length (e.g. 64) to help it "remember" the primed style across more virtual targets.
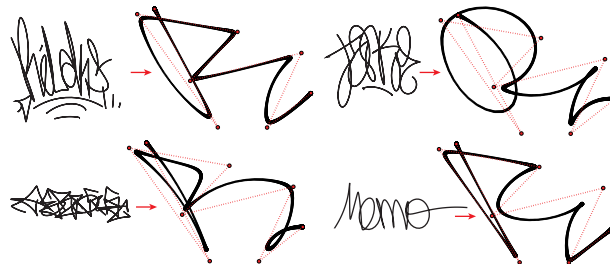

Figure 10: Dynamic parameters generated over user specified virtual targets (in red) using *a single model* trained on 4 examples (with data augmentation x2000). Each trajectory has been generated by priming the network with the corresponding example. Training examples: top row are from the GML database, bottom row were drawn using a tablet.

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 (Fig. 11).

Figure 11: Variations generated by varying the random number generator seed prior to sampling.

4.2 Trajectory stylisation.

We are also able to apply these methods to input paths previously drawn by a user or taken from an existing online dataset. Given such an input path, first we use our $\Sigma\Lambda$ parameter reconstruction method to extract a series of virtual targets and corresponding dynamic parameters that reconstruct the input path. We then *discard* the reconstructed dynamic parameters and replace them with the ones predicted by the same DPP models we used in the previous examples. The result is an output that is structurally similar to the original path, but possesses the dynamic and geometric features of another.
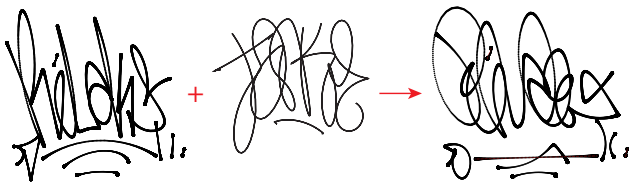

Figure 12: Less satisfactory case for the stylisation of a complex tag. Left: user drawn trace to be stylised, Middle: training example, Right: Output of the model, applying the style of the training example to the user drawn trace.
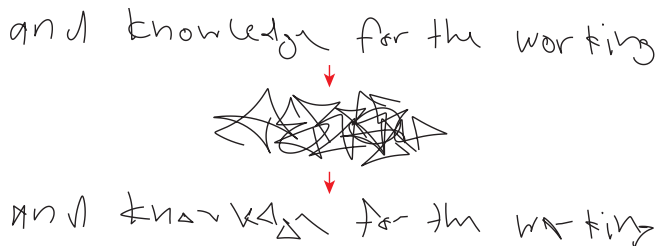

Figure 13: Stylisation on longer sequences. Top: a sequence from the IAM dataset [23]. Middle: single user-specified sample used to train a DPP model. Bottom: final output.

We test the method with a simple sequence of letters drawn by a user with a tablet and observe that, depending on the example used to prime the network, the method produces clearly different and readable stylisations of the input (Fig. 1). 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 fact, with a more complex example, the original readability of the input may be lost and the quality of the resulting stylisation is not satisfactory, or in general difficult to evaluate qualitatively (Fig. 12). We also try this method on longer sequences of user-drawn traces such as those found in the IAM dataset [23] (Fig. 13).

4.3 Fully generative traces

Finally, we explore the application of our method in a generative pipeline in which a model learns and generates virtual target sequences, while the final trajectory and different stylisations are determined by the DPP model itself. For this task, we train a *Virtual Target Prediction* (VTP) model, which builds from the handwriting prediction model originally developed by Graves [17]. However, our VTP model implementation operates on *sparse sequences of virtual targets* rather than dense raw point sequences. Due to this sparsity, it is then possible to augment the input in a coherent manner by perturbing the virtual targets positions with random offsets relative to the sequence extent. As a result, we can also train the model from a very small number of samples.

The process is as follows: training is performed on a first VTP model from virtual target sequences extracted from a small number of user made graffiti examples (e.g. between 1 and 4); we then use these models in combination to generate new virtual target sequences and patterns that are similar to the ones given in the training set. By priming the VTP model with a desired virtual target style example, and priming the DPP model with a desired dynamic-parameter style example (as in Fig. 10), it is then possible to control and mix the overall style of the generated trace (Fig. 14). This process can also be directly used with large input datasets; e.g. we have trained VTP models on larger numbers of targets extracted from the IAM online handwriting dataset [23] (as originally used by Graves [17]).
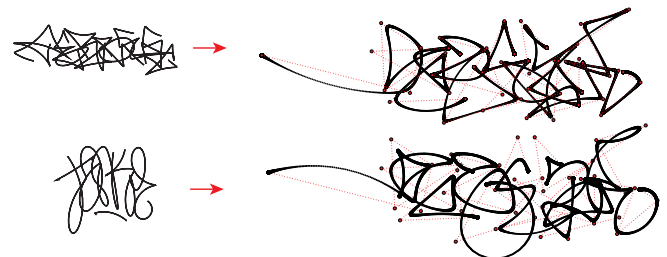

Figure 14: Generating asemic graffiti. One network predicts sequences of virtual targets based on a small number of examples given by a user. Our DPP models are then used to predict the dynamics, resulting in different stylizations.

5 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. [21, 26]). However, we propose a movement based approach

to stylisation, 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. For example, the output of our method reflects a realistic and natural movement which can be used to (i) produce more expressive renderings of the trajectories by generating realistic stroke animations or (ii) drive the natural motions of an animated character or robotic drawing device. Furthermore, the variation of model parameters can be used (as demonstrated) to augment training data, but also to generate realistic variations in the generated outputs.

Our method currently relies on an accurate reconstruction of the input in the preprocessing step. Improvements should target especially parts that depend on user tuned parameters, such as the identification of salient points along the input trace (which requires a final peak detection pass), and measuring the sharpness of the input in correspondence with salient points.

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 [17], with the additional benefits of resolution independence and the possibility of training on a much reduced dataset size, even achieving satisfying results with one-shot learning situations.

ACKNOWLEDGMENTS

REFERENCES

[1] J. Bergstra and Y. Bengio. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* 13 (2012), 281–305.
[2] D. Berio, S. Calinon, and F. Fol Leymarie. 2016. Learning dynamic graffiti strokes with a compliant robot. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, 3981–6. https://doi.org/10.1109/IROS.2016.7759586.
[3] D Berio, S. Calinon, and F. Fol Leymarie. 2017. Dynamic Graffiti Stylisation with Stochastic Optimal Control. In *ACM Proceedings of the 4th International Conference on Movement and Computing*. London, UK.
[4] D Berio, S. Calinon, and F. Fol Leymarie. 2017. Generating Calligraphic Trajectories with Model Predictive Control. In *Proceedings of Graphics Interface*. Canadian Human-Computer Communications Society, Edmonton, Canada.
[5] D. Berio and F. Fol Leymarie. 2015. Computational Models for the Analysis and Synthesis of Graffiti Tag Strokes. In *Proceedings of the Workshop on Computational Aesthetics*. Eurographics Association, 35–47.
[6] H. Bezine, A.M. Alimi, and N. Sherkat. 2004. Generation and analysis of handwriting script with the Beta-elliptic model. *Proc. of International Workshop on Frontiers in Handwriting Recognition (IWFHR)* 8, 2 (2004), 515–20.
[7] C.M. Bishop. 1994. Mixture density networks. (1994). Technical Report NCRG/4288, Neural Computing Research Group, Aston University, U.K.
[8] L. Crnkovic-Friis and L. Crnkovic-Friis. 2016. Generative Choreography using Deep Learning. *arXiv preprint arXiv:1605.06921* (2016).
[9] Kingma D. and Ba J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014), 1–13.
[10] J. De Winter and J. Wagemans. 2008. Perceptual saliency of points along the contour of everyday objects: A large-scale study. *Perception & Psychophysics* 70, 1 (2008), 50–64.
[11] A. Dempster, N. Laird, and D. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Royal Statistical Society B* 39, 1 (1977), 1–38.
[12] J. Feldman and M. Singh. 2005. Information along contours and object boundaries. *Psychological review* 112, 1 (2005), 243.
[13] A. Fischer, R. Plamondon, C. O'Reilly, and Y. Savaria. 2014. Neuromuscular Representation and Synthetic Generation of Handwritten Whiteboard Notes. In *Proc. of Frontiers in Handwriting Recognition (ICFHR)*. 222–7.
[14] T. Flash and A.A. Handzel. 2007. Affine differential geometry analysis of human arm movements. *Biological cybernetics* 96, 6 (2007), 577–601.
[15] T. Flash and B. Hochner. 2005. Motor primitives in vertebrates and invertebrates. *Current opinion in neurobiology* 15, 6 (2005), 660–6.
[16] F.A. Gers, J. Schmidhuber, and F. Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation* 12, 10 (2000), 2451–2471.
[17] A. Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).
[18] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. 2015. DRAW: A Recurrent Neural Network For Image Generation. *arXiv preprint arXiv:1502.04623* (2015).
[19] D. Ha. 2015. Generative Handwriting Demo using TensorFlow. (2015). https://github.com/hardmaru/write-rnn-tensorflow
[20] D. Ha, A. Dai, and Q.V. Le. 2016. HyperNetworks. *arXiv preprint arXiv:1609.09106* (2016).
[21] A. Hertzmann, N. Oliver, B. Curless, and S.M. Seitz. 2002. Curve Analogies. In *Proc. 13th EuroGraphics Workshop on Rendering (EGRW)*. Pisa, Italy, 233–46.
[22] S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–80.
[23] E. Indermühle, M. Liwicki, and H. Bunke. 2010. IAMonDo-database: An Online Handwritten Document Database with Non-uniform Contents. In *Proc. 9th IAPR Int'l Workshop on Document Analysis Systems (DAS)*. 97–104.
[24] H.S.R. Kao, R. Hoosain, and G.P. Van Galen. 1986. *Graphonomics: Contemporary research in handwriting*. Elsevier.
[25] Graffiti Research Lab. 2009. Graffiti Analysis Database (000000book.com). Web. (2009). http://000000book.com
[26] K. Lang and M. Alexa. 2015. The Markov pen: Online synthesis of free-hand drawing styles. In *Proc. of the workshop on Non-Photorealistic Animation and Rendering (NPAR)*. Eurographics Association, 203–15.
[27] H.W. Lin and M. Tegmark. 2016. Why does deep and cheap learning work so well? *arXiv preprint arXiv:1608.08225* (2016).
[28] M. Ltaief, H. Bezine, and A.M. Alimi. 2012. A neuro-Beta-elliptic model for handwriting generation movements. In *Proc. of International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 803–8.
[29] H. Nagasaki. 1989. Asymmetric velocity and acceleration profiles of human arm movements. *Experimental Brain Research* 74, 2 (1989), 319–26.
[30] V. Nair and G. Hinton. 2005. Inferring motor programs from images of handwritten digits. In *Advances in neural information processing systems*. 515–22.
[31] C. O'Reilly and R. Plamondon. 2008. Automatic extraction of Sigma-Lognormal parameters on signatures. In *Proc. of 11th International Conference on Frontier in Handwriting Recognition (ICFHR)*.
[32] R. Pascanu, T. Mikolov, and Y. Bengio. 2013. On the difficulty of training Recurrent Neural Networks. In *Proc. of ICML*, Vol. 28. 1310–8.
[33] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. 2014. Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In *Proc. of ICFHR*. IEEE, 285–90.
[34] A. Pignocchi. 2010. How the Intentions of the Draftsman Shape Perception of a Drawing. *Consciousness and Cognition* 19, 4 (2010), 887–98.
[35] R. Plamondon. 1995. A Kinematic Theory of Rapid Human Movements. Part I . Movement Representation and Generation. *Biological cybernetics* 72, 4 (1995), 295–307.
[36] R. Plamondon and others. 1993. Modelling velocity profiles of rapid movements: A comparative study. *Biological cybernetics* 69, 2 (1993), 119–28.
[37] R. Plamondon and others. 2014. Recent developments in the study of rapid human movements with the kinematic theory. *Pattern Recognition Letters* 35 (2014), 225–35.
[38] R. Plamondon and C. Privitera. 1996. A neural model for generating and learning a rapid movement sequence. *Biological cybernetics* 74, 2 (1996), 117–30.
[39] B. Rohrer and N. Hogan. 2006. Avoiding spurious submovement decompositions II. *Biological cybernetics* 94, 5 (2006), 409–14.
[40] D.A. Rosenbaum and others. 1995. Planning reaches by evaluating stored postures. *Psychological Review* 102, 1 (1995), 28–67.
[41] M. Schuster. 1999. Better Generative Models for Sequential Data Problems: Bidirectional Recurrent Mixture Density Networks. In *Proc. of NIPS*. 589–95.
[42] I. Sutskever. 2013. *Training Recurrent Neural Networks*. Ph.D. Dissertation. University of Toronto.
[43] M. Umilta and others. 2012. Abstract Art and Cortical Motor Activation: An EEG Study. *Frontiers in Human Neuroscience* 6, 311 (2012), 1–9.
[44] T. Varga, D. Kilchhofer, and H. Bunke. 2005. Template-based Synthetic Handwriting Generation for the Training of Recognition Systems. In *Proc. of 12th Conf. of the International Graphonomics Society (IGS)*. 206–11.
[45] X.-Y. Zhang and others. 2016. Drawing and Recognizing Chinese Characters with Recurrent Neural Network. *arXiv preprint arXiv:1606.06539* (2016).