# Dynamic Graffiti Stylisation with Stochastic Optimal Control

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

Sylvain Calinon
Idiap Research Institute
sylvain.calinon@idiap.ch

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

## ABSTRACT

We present a method for the interactive generation of stylised letters, curves and motion paths that are similar to the ones that can be observed in art forms such as graffiti and calligraphy. We define various stylisations of a letter form over a common geometrical structure, which is given by the spatial layout of a sparse sequence of targets. Different stylisations are then generated by optimising the trajectories of a dynamical system that tracks the target sequence. The evolution of the dynamical system is computed with a stochastic formulation of optimal control, in which each target is defined probabilistically as a multivariate Gaussian. The covariance of each Gaussian explicitly defines the variability as well as the curvilinear evolution of trajectory segments. Given this probabilistic formulation, the optimisation procedure results in a *trajectory distribution* rather than a single path. It is then possible to stochastically sample from the distribution an infinite number of dynamically and aesthetically consistent trajectories which mimic the variability that is typically observed in human drawing or writing. We further demonstrate how this system can be used together with a simple user interface in order to explore different stylisations of interactively or procedurally defined letters.

## CCS CONCEPTS

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

## KEYWORDS

Human hand-writing movement modeling; procedural calligraphy; graffiti and tags generation; model predictive control; stochastic optimal control; smoothing splines; iconic and kinemic letter forms.

**ACM Reference format:**
Daniel Berio, Sylvain Calinon, and Frederic Fol Leymarie. 2017. Dynamic Graffiti Stylisation with Stochastic Optimal Control. In *Proceedings of MOCO conference, London, United Kingdom, 2017 (MOCO),* 8 pages.
DOI: 10.475/123_4

**Figure 1: Variations on a tag. (a) In (1) is an original tag made with a marker by a graffiti artist; (2) a user rapidly sketches a control polygon by placing points (targets) near curvature extrema in overlay working from an image of the tag; (3) the user adjusts the Gaussians interactively to follow the trace of the original tag; (4) the reproduced tag rendered with a textured brush. (b) Variations on the tag by modifying the parameter $d$. (c) Variations on the tag generated from the specified targets using semi-tied covariances (illustrated as orientable yellow ellipses).**

## 1 INTRODUCTION

In this study we adopt tools from optimal control, robotics and computational motor control to generate synthetic traces that are visually and dynamically similar to the ones made by a human expert when drawing or writing. We describe a system that enables a user or an algorithm to rapidly define such traces through the specification of a control polygon made of a coarse sequence of targets. The user can then generate and interactively manipulate a family of motion trajectories, which follow the target layout and are characterised by dynamics that are similar to the ones that can typically be observed in human hand movements (Fig. 1). The smooth dynamics produced by the system can be exploited to generate natural looking stroke animations, expressive renderings of the trajectory evolution, or drive the smooth end effector motions of a robotic drawing device. In this study we particularly emphasise the applications of our system to the generation of traces that mimic the visual quality of certain forms of calligraphy and graffiti art.

Graffiti, which is also commonly referred to as "writing" or "aerosol art", is an art form that emerged in the late 1960s when it started to appear on the surfaces of the New York City subway [14, 28]. Since then graffiti has developed into a rich and complex art form that revolves around various stylisation and abstractions applied to the letters of an alphabet, and that can be seen today on the walls and surfaces of most urbanscapes around the globe. *Movement* plays two different roles in graffiti. On the one hand, it

*MOCO, London, United Kingdom*

is in itself at the source of concern of this contemporary art form, where the surfaces of subway trains serve the role of a *moving canvas*. Graffiti art is also often seen along train tracks and on the walls bordering highways and busy roads, where it is meant to be appreciated while the observer is moving. On the other hand, the mastery of rapid hand gestures is crucial to the aesthetics and style obtained in producing the traces forming the artefacts [19, 36]. In our work we are mainly concerned with the latter aspect involving movement and propose a probabilistic computational framework to model the production of graffiti.

Furthermore, in this study, we focus on the earliest and fundamental type of graffiti art: the highly stylised signature of an artist's pseudonym, commonly referred to as a *tag*. The manner in which a tag is written is commonly referred to as "handstyle" [14] and identifies the artists's personal style and skill. A well executed handstyle is the result of years of practice, and its visual quality is directly related to the spontaneity in which the movements are executed. This is reflected in graffiti jargon with the term *flow*, which denotes the quality of execution of a tag.

We consider the graffiti stylisation of a letter-form with an approach inspired by the work of semiotician *William C. Watt* [47] who studied the evolution of the Latin alphabet with two complementary descriptions of the letter form: one *iconic* where the letter is described in its basic structure as a *sign*, and one *kinemic* — the study of gestures as body language — where the letter is considered as a dynamic representation of the movements that produce its trace on canvas. As an example, Watt demonstrated how the same iconic representation transforms an upper case "A" into a lower case "$\alpha$" through a process he calls *facilitation*, which is the tendency to reduce effort during the kinemic production of a letter. In our study we explore a similar approach for the synthetic generation of different graffiti handstyles. We define an iconic description of a letter form through a coarse sequence of target loci: the centres of multivariate Gaussian distributions with full covariances. The kinemic realisation of the letter is then produced using a stochastic optimal control formulation, in which a dynamical system is optimised to follow the spatial layout of the targets as well as the coordination patterns defined by the covariances.

With such a probabilistic formulation, the optimisation process results in a ***distribution*** of trajectories [8], rather than a single path. This allows for example to easily capture the subtle variations that can be typically observed in multiple instances of writing or drawing by the same person (§3.6). In addition, varying the shapes of the Gaussians as well as modifying the optimisation and dynamical system parameters, result in different kinemic realisations of the same target sequence (Fig. 1.(c)). This in turn generates different trajectories that are qualitatively similar to different handstyles that can typically be observed in graffiti tags produced by a human artist.

The rest of this paper is organised as follows: after a brief background on related work (§2), we will first provide a detailed description of the optimal control method used to generate trajectories (§3) and then demonstrate how it can be applied for the procedural stylisation (§4) and generation (§4.2) of graffiti tags.

## 2 BACKGROUND

A rich history of experimental research has brought to light a number of principles that characterise human hand motions, based on dynamic (time, speed) and figural (curvature, shape) aspects. The tangential speed profile of point-to-point aiming movements typically assumes a "bell shape" [17, 33, 38], variably asymmetric depending on the rapidity of the movement [35, 39]. It is generally accepted that complex movements can be described with the superimposition of a discrete number of basic "ballistic" primitives often referred to as *strokes* [34, 41, 43], which are also characterised by bell shaped velocity profiles. With experience, a movement tends to become smoother [39–41] and the number of velocity peaks decreases. This phenomenon is known as *co-articulation* and can be interpreted as the *chunking* of movement primitives at the planning level [41]. The speed of human hand movements tends to be inversely proportional to the trajectory curvature [16, 20]; in certain types of movement, this relation takes the form of a power law [30, 46]. The duration of each movement primitive tends to be similar and independent of the whole movement extent, a phenomenon referred to as local isochrony [27].

Hand movements are typically smooth and appear to obey optimality principles based on the magnitude of high order derivatives of position, leading to various proposed optimisation computational approaches minimising, for example: variance [24], torque [45], "jerk" (or 3rd order derivatives) [18], "snap" (4th order) [15]. In such models, the evolution of a movement is typically defined with point loci along the trajectory that function similarly to spline interpolation points, and which are commonly referred to as *via-points*. A number of models explicitly describe complex motions with the space-time superimposition of ballistic stroke primitives, where the speed profile of a stroke follows a specified bell shaped function, such as a lognormal [37, 38] or a beta function [7]. In such cases, the trajectory evolution is described with a sequence of positions that do not strictly lie along the rendered trajectory, but rather describe the aiming targets of consecutive strokes; these loci are comparable to the control points of a smoothing spline and are commonly called *virtual targets*. Our trajectory description method can be seen as a hybrid between via-points and virtual targets, where Gaussians with low variance behave similarly to via-points, while those with high variance are alike virtual targets.

Egerstedt and Martin [12] use a Hilbert space representation to show that Bézier curves, splines and smoothing splines can be interpreted as solutions to an optimal control problem. In our approach we also solve an optimal control problem by formulating tracking costs as full precision matrices. As such, we may interpret our trajectory generation method as an extension of smoothing splines encapsulating information about precision, coordination and dynamics.

An important number of projects in computer graphics have considered the generation of artistic imagery [29] and the stylisation of line drawings [21, 25, 31]. However, very few such works have exploited motion synthesis techniques; we highlight some of these next. Haeberli [23] created a program that generates calligraphic stylisations of a computer mouse trace on the basis of a mass-spring system. House et al. [26] generate sketchy renderings of a 3D model by using a Proportional Integral Derivative (PID) controller. AlMeraj et al. [1] mimic the undulation of hand drawn pencil lines by using the minimum jerk model. Berio and Leymarie [6] use the sigma lognormal model [38] to interactively define the motion paths and variations of graffiti tag trajectories.

In a recent companion paper [5], we describe how our method is suitable for interactive applications similar to the popular computer aided design techniques such as Bézier curves and splines; we also provide a more detailed overview of the implementation based on the principles of stochastic optimal control. In the work reported here, we extend the latter method towards generative applications, and focus on the task of trajectory *stylization* inspired in particular by the work of W. C. Watt [47]. We also introduce the use of semi-tied covariances to allow the user to rapidly explore different stylisations in an intuitive manner (Fig. 1.(c)).

# 3 TRAJECTORY GENERATION

We describe a trajectory by optimising the evolution of a dynamical system controlled by its highest order derivative along the spatial layout of an ordered sequence of multivariate Gaussians. The optimisation is formulated with a cost function that forces the system to track the Gaussians while limiting the amplitude of the control command. The resulting trajectory is smooth up to the order of the dynamical system, and the corresponding dynamics are similar to the ones that would be seen in a movement made by a drawing hand, with desirable features such as bell shaped speed profiles and an inverse relation between speed and curvature. The centres of the Gaussians define a form of control polygon or "motor plan" (in robotics' jargon) that describe the overall spatial evolution of the trajectory. Also, the covariances permit to define the variability as well as directional trends of trajectory segments (Fig. 2).

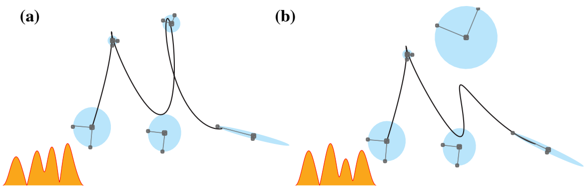


**Figure 2: Examples of Gaussian targets (4th order system) with associated speed profiles (bottom left), one (bell shaped) per stroke. Note that in (b) the top variance has been increased, which facilitates the evolution of the trajectory into the slim and elongated covariance at the bottom right.**

As previously mentioned, this representation can be seen as a hybrid between the traditionally used via-point and virtual target trajectory descriptors: with a low variance, the trajectory is forced to pass close to the mean of the distribution, which effectively results in a close approximation of a via-point. Using a higher variance produces an effect similar to smoothing splines, with the centers of the Gaussians acting as virtual targets. In addition, non-spherical covariances allow to capture more complex spatial constraints, such as forcing a movement to follow a given direction or to pass through a narrow region of space (Fig. 2). The behaviour of such a system is consistent with the *minimal intervention principle* [44, 48], which proposes that deviations from an average trajectory are only corrected when they interfere with the required precision of a task. In our case, we locally achieve the required precision by tuning Gaussian covariances.

## 3.1 Dynamical system

We generate a trajectory with an $n$th order discrete linear time invariant system defined with the state space form:

$$\xi_{t+1} = \boldsymbol{A}\xi_t + \boldsymbol{B}\boldsymbol{u}_t, \tag{1}$$

where the state

$$\xi_t = \left[\boldsymbol{x}_t^\top, \dot{\boldsymbol{x}}_t^\top, \ldots, \overset{(n-2)}{\boldsymbol{x}_t}^\top, \overset{(n-1)}{\boldsymbol{x}_t}^\top\right]^\top \tag{2}$$

contains the position and its derivatives up to order $n-1$, and the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ describe the time invariant response of the system to an input command $\boldsymbol{u}_t$. For the examples presented here, we utilise a chain of $n$ integrators commanded by its highest derivative, with (continuous) system matrices:

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix}, \quad \bar{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix}. \tag{3}$$

The discrete time versions of the system matrices can be computed by using a Zero Order Hold (ZOH) or a simple forward Euler discretisation given by:

$$\boldsymbol{A} = \Delta t \bar{\boldsymbol{A}} + \boldsymbol{I} \quad \text{and} \quad \boldsymbol{B} = \Delta t \bar{\boldsymbol{B}}. \tag{4}$$

## 3.2 Optimisation

An optimal trajectory of $N$ time steps is computed by minimising a tradeoff between deviations from a desired reference state $\hat{\xi}_t$ (*tracking cost*) and limiting the magnitude of the control commands (*control cost*) with a quadratic cost function:

$$J = \sum_{t=1}^{N} \left(\hat{\xi}_t - \xi_t\right)^\top \boldsymbol{Q}_t \left(\hat{\xi}_t - \xi_t\right) + \sum_{t=1}^{N-1} \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t, \tag{5}$$

where $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ are positive semi-definite weight matrices that define the tradeoff between tracking and control penalties for each time step.

This type of optimisation problem is commonly used in process control and robotics applications, where it is known as discrete Linear Quadratic Tracking (dLQT) corresponding to the linear unconstrained case of Model Predictive Control (MPC) [49]. In a typical control setting, these methods would be used to compute an optimal control command for the current time step based on a linearization of the system, and then repeated iteratively for the subsequent time steps. The mathematical framework is however more general and can also be exploited within a planning perspective. In our trajectory synthesis use case, by assuming a system with no disturbance, we can compute all the commands in a single batch optimisation step and generate the resulting trajectory in a rapid manner.

### 3.3 Tracking cost

We describe a trajectory with an ordered sequence of $m$ multivariate Gaussians $\left\{\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right\}_{i=1}^{m}$, each defining a state. With an assumption of *local isochrony*, we keep a fixed duration $T_s$ per state, which gives a total trajectory duration of $T = mT_s$ and a corresponding discretised trajectory with $N = T/\Delta_t$ time steps. The tracking weights $\boldsymbol{Q}_t$ and target $\hat{\xi}_t$ for each time step are then formulated by repeating each state $T_s/\Delta_t$ times in a stepwise manner along a state vector $\boldsymbol{s} \in \mathbb{N}^N$ (*e.g.* $\boldsymbol{s} = \{1, 1, 2, 2, 2, 3, \ldots, m\}$ ) and letting:

$$\hat{\xi}_t = \boldsymbol{\mu}_{s_1} \quad \text{and} \quad \boldsymbol{Q}_t = \boldsymbol{C}^{\top} \Sigma_{s_t}^{-1} \boldsymbol{C}, \tag{6}$$

with a *sensor* matrix:

$$\boldsymbol{C} = [\boldsymbol{I}, \boldsymbol{0}, \ldots, \boldsymbol{0}], \tag{7}$$

producing zero entries in $\boldsymbol{Q}_t$ for the state derivative terms. This corresponds to a feedback system observing only positions and allowing the specification of states only using position constraints.

This probabilistic formulation of the system states lends itself well for being manipulated interactively or procedurally in an interface that is similar to conventional curve editing techniques such as Bézier curves or splines [5]. A natural interface is then to let the user manipulate an ellipsoid, the axes of which map to the corresponding covariances (Fig. 2). Such covariances can be generated through the eigendecomposition:

$$\Sigma_i = \Theta_i S_i \Theta_i^{\top}, \tag{8}$$

where $\Theta_i$ and $S_i^{\frac{1}{2}}$ correspond in the interface to the rotation and scaling matrices defined by the ellipsoid axes.

### 3.4 Control cost

The weight matrices $\boldsymbol{R}_t$ define a penalty on the amplitude of control commands. Typically this cost is formulated as a constant diagonal term that is inversely proportionally to the maximum square norm of the control command. In order to achieve approximately equal tracking performance across different system orders (Fig. 3), we express the control cost in terms of a *maximum allowed displacement* $d$, and compute $\boldsymbol{R}_t$ using the frequency gain of the integrator chain:

$$\boldsymbol{R}_t = \frac{1}{(\omega^n d)^2} \boldsymbol{I} \quad \text{and} \quad \omega = 2\pi T_s, \tag{9}$$

where the frequency, $\omega$, is empirically set using the state duration as a period. Lower values of $d$ tend to smooth the trajectory, while higher values generate sharper paths. Because the cost function is defined as a tradeoff between tracking and control cost, it is in practice possible to achieve the same effect by either increasing the variance of the Gaussians or decreasing the value of $d$.

### 3.5 Least squares solution

The optimal trajectory can be retrieved iteratively using dynamic programming [5, 9], or in batch form by solving a large regularised least squares problem. Here we describe the latter, which is more compact and allows a straightforward probabilistic interpretation of the result. To compute the least squares solution, we exploit the time invariance of the system, and express all future states as a function of the initial state $\xi_1$ with:

$$\xi = \boldsymbol{S}_{\xi} \xi_1 + \boldsymbol{S}_{\boldsymbol{u}} \boldsymbol{u}, \tag{10}$$

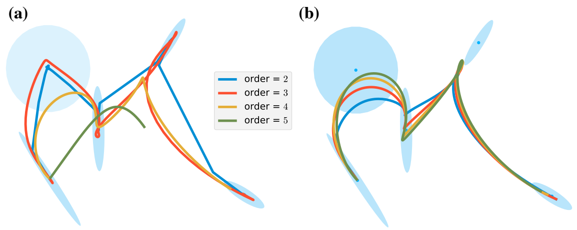


**Figure 3: (a) Trajectories with $R_t$ constant across increasing orders of the dynamical system. (b) Improved tracking consistency by computing $R_t$ depending on the system order and a maximum displacement parameter $d$.**

where

$$\boldsymbol{S}_{\xi} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix} \quad \text{and} \quad \boldsymbol{S}_{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{AB} & \boldsymbol{B} & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \ldots & \boldsymbol{B} \end{bmatrix}. \tag{11}$$

We then express the cost function (5) in matrix form as:

$$J = (\hat{\xi} - \xi)^{\top} \boldsymbol{Q} (\hat{\xi} - \xi) + \boldsymbol{u}^{\top} \boldsymbol{R} \boldsymbol{u}, \tag{12}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are large diagonal block matrices (with $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ as diagonal block elements), while $\hat{\xi}$, $\xi$ and $\boldsymbol{u}$ are column vectors respectively stacking the reference, state and control commands. Substituting (10) into (12), differentiating with respect to $\boldsymbol{u}$ and setting to zero results in a regularized least squares solution for the command sequence gives:

$$\boldsymbol{u} = \underbrace{(\boldsymbol{S}_{\boldsymbol{u}}^{\top} \boldsymbol{Q} \boldsymbol{S}_{\boldsymbol{u}} + \boldsymbol{R})^{-1}}_{\Sigma_{\boldsymbol{u}}} \boldsymbol{S}_{\boldsymbol{u}}^{\top} \boldsymbol{Q} \left(\hat{\xi} - \boldsymbol{S}_{\xi} \xi_1\right), \tag{13}$$

which is then substituted back into (10) to generate a trajectory. From (13) we can see that $\boldsymbol{R}$ effectively acts as a Tikhonov regularisation term (aka ridge regression or weight decay) in the least squares solution, resulting in a smoothing effect on the generated trajectory.

### 3.6 Stochastic sampling

Because the cost function (12) is a sum of square error terms, its minimisation can be interpreted probabilistically as the product of two Gaussians:

$$\mathcal{N}(\boldsymbol{u}, \Sigma_{\boldsymbol{u}}) \sim \mathcal{N}\left(\boldsymbol{S}_{\boldsymbol{u}}^{-1}(\hat{\xi} - \boldsymbol{S}_{\xi}\xi_1), \boldsymbol{S}_{\boldsymbol{u}}^{\top} \boldsymbol{Q} \boldsymbol{S}_{\boldsymbol{u}}\right) \times \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}), \tag{14}$$

which describes a distribution of control commands with center $\boldsymbol{u}$ and covariance $\Sigma_{\boldsymbol{u}}$. By using the linear relation (10), the distribution in control space can be converted to a *trajectory distribution* (refer to the article by Calinon [8] for details):

$$\mathcal{N}\left(\xi, \Sigma_{\xi}\right) \quad \text{with} \quad \Sigma_{\xi} = \boldsymbol{S}_{\boldsymbol{u}} \Sigma_{\boldsymbol{u}} \boldsymbol{S}_{\boldsymbol{u}}^{\top}. \tag{15}$$

Such a distribution can be used to generate natural variations around the average trajectory $\xi$ (Fig. 4) with:

$$\xi \sim \boldsymbol{\mu}_{\xi} + \boldsymbol{V}_{\xi} \Lambda_{\xi}^{\frac{1}{2}} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{16}$$

computed from the eigendecomposition:

$$\Sigma_\xi = V_\xi \Lambda_\xi V_\xi^{\top}, \tag{17}$$

where $V_\xi$ is a matrix containing the eigenvectors and $\Lambda_\xi$ is a diagonal matrix containing the eigenvalues.

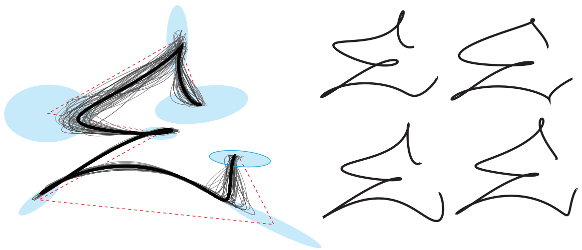

**Figure 4: Stochastic sampling of the trajectory distribution. *Left:* the main trajectory (in black) with random samples from the trajectory distribution (in gray) and the corresponding Gaussians. *Right:* randomly selected samples from the same trajectory distribution.**

### 3.7 Multiple references

The output of the dLQR optimisation procedure can be interpreted as a time varying flow field that depends on the minimisation of the tracking term of the cost function (12). Such a formulation can be extended to additional quadratic costs. This is exploited for example in a robot learning by demonstration application by Calinon [8] to express the cost function in $P$ different coordinate systems. The cost function (12) then becomes:

$$J = \sum_{i=1}^{P} (\hat{\xi}_i - \xi)^{\top} Q_i (\hat{\xi}_i - \xi) + u^{\top} R u \tag{18}$$

and a trajectory of control commands can be retrieved with:

$$u = \left( \sum_{i=1}^{P} S_u^{\top} Q_i S_u + R \right)^{-1} \sum_{i=1}^{P} S_u^{\top} Q_i \left( \hat{\xi}_i - S_\xi \xi_1 \right). \tag{19}$$

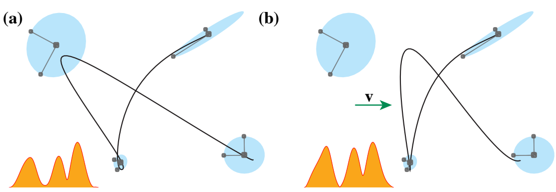


**Figure 5: Additional velocity reference (3rd order system). (a) Trajectory tracking four Gaussians. (b) Trajectory generated with an additional velocity reference ($\boldsymbol{v}$). Note that the optimisation prioritises the Gaussian with low variance, in agreement with the minimum intervention principle [44].**

In addition to the position tracking constraints (Fig. 5a), we can add penalties on the velocity terms, such as forcing the trajectory

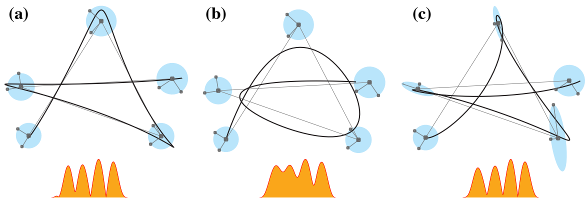


**Figure 6: (a) 4th order system following a sequence of 5 targets with spherical covariance. (b) *Facilitation* [47] using the same system with a lower value of $d$. (c) Effect of non spherical covariance. The respective speed profiles are shown below each trajectory.**

velocity to be under the influence of a constraint vector $\boldsymbol{v}$ by using two references ($P = 2$), where the first tracking constraint is defined similarly as in the previous examples, and where a second tracking constraint is given by

$$\hat{\xi}_{2,t} = \mathbf{0} \quad \text{and} \quad Q_{2,t} = C_{\boldsymbol{v}}^{\top} \Sigma_{\boldsymbol{v}} C_{\boldsymbol{v}} \quad \forall t \in \{1, \ldots, N\}, \tag{20}$$

with

$$\Sigma_{\boldsymbol{v}} = \boldsymbol{v}\boldsymbol{v}^{\top} \quad \text{and} \quad C_{\boldsymbol{v}} = [\mathbf{0}, \boldsymbol{I}, \mathbf{0}, \ldots, \mathbf{0}], \tag{21}$$

in which case the sensor matrix $C_{\boldsymbol{v}}$ corresponds to a feedback system in which only velocity is observed (Fig. 5b).

## 4 MIMICKING HANDSTYLES

In this section we explore the application of a dual representation of letter form related to the one proposed by Watt [47], in order to mimic computationally the aesthetic and dynamics of graffiti handstyles. In his original formulation, Watt uses a formal grammar to describe the *iconic* representation of a letter in an order/time independent manner. Here, for implementation convenience, we opt for a different approach in which the iconic description of the letter is given by the *centers of a sequence of Gaussians* (or *target sequence*), which consequently also describes a temporal ordering of strokes. The corresponding *kinemic* representation is then given by the covariances associated with each Gaussian as well as the remaining optimisation and dynamical system parameters. The variation of these parameters results in different kinemic realisations of the same target sequence, which we adopt in this study as a *computational* definition of *graffiti handstyle*.

We describe the structure of a letter with sequences of targets, one for each part of the letter made with the pen touching the canvas. Each sequence is made of a low number of targets (usually between 2 and 9), each corresponding to a Gaussian, and can be easily sketched by a user in a point and click procedure. A trajectory generated over a sequence of $m$ targets will typically produce a speed profile characterised by $m - 1$ peaks and (consistently with the inverse speed/curvature relation) local minima corresponding with curvature extrema along the trajectory. To produce a minimal target sequence from an existing letter trajectory, we use an inverse approach and manually (Fig. 1) or automatically (Fig. 10) place targets near salient positions such as curvature extrema [2, 6, 13].

As a simple example of our approach, we mimic the process of facilitation described by Watt. This can be done by specifying spherical covariances for each target and then varying the maximum displacement parameter $d$. With a larger value of $d$ the trajectory closely follows the layout of the targets, resulting in a capital "A" (Fig. 6a), while a lower value of $d$ produces a co-articulation effect that produces a trace which is similar to a lower case "a" (Fig. 6b). If we introduce non-isotropic (full) covariances, we can observe that the resulting trajectory follows a more complex and "calligraphic" like evolution (Fig. 6c), which is evocative of effects that can be seen in calligraphy and graffiti produced by humans.

The trajectories generated by our system are sequences of points, the resolution of which depends on the discretisation time step ($\Delta t$). The distance between consecutive points is not constant and reflects the smooth and physiologically plausible dynamics generated by the model. As a result, it is trivial to generate natural looking stroke animations by incrementally sweeping a brush texture along the points of the trajectory. To increase the sense of dynamism, we slightly vary the brush size at a degree inversely proportional to the trajectory speed, which mimics the effect of more ink being deposited on a surface when the movement is slower (Fig. 1).

## 4.1 Semi-tied covariances

In the previous section, we have seen that it is possible for a user to easily edit the shape and position of each Gaussian, where variations of the covariance shape and size result in different kinemic realisations and stylisations of the target sequence. For more generative-oriented applications, it is desirable to formulate a more parsimonious way of generating trajectories that are consistently similar to diverse graffiti handstyles.

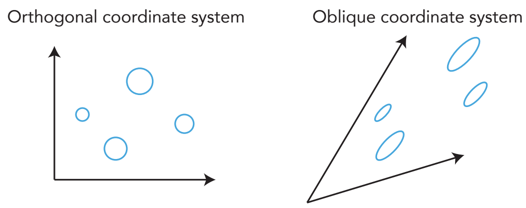


**Figure 7: Effect of an oblique coordinate system (right) on the covariances (in turquoise).**

In our experiments we observe that one possible way to achieve this result is to enforce a *shared orientation* for all covariance ellipsoids. This is known as *semi-tied* covariances and it corresponds to the case of shared eigenvectors but not necessarily with same eigenvalues. This can be interpreted as the alignment of different movement parts/primitives with a shared coordination pattern [42], which is in line with the hypothesis of postural-synergies at the motor planning level [10]. The tied formalism implies a shared non-orthogonal (oblique) basis for all the covariances. This produces a shear transformation that in the 2D case transforms a circle into an oriented ellipse (Fig. 7). Oblique coordinates have also been suggested to describe the coordination of handwriting movements made with the fingers and wrist [11], which suggests another possible bio-physical interpretation of this result.

In order to manipulate and edit semi-tied covariances, we define a oblique basis $\boldsymbol{H}$ and a magnitude $h$. The shared covariances can

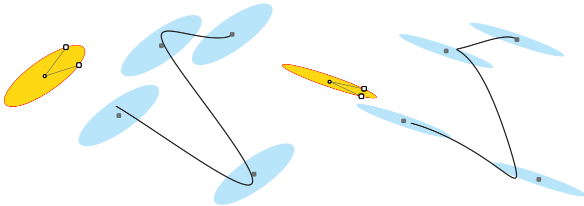

**Figure 8: Interface for manipulating semi-tied covariances and corresponding trajectories. The user can drag at the border of the yellow ellipsoid the pair of small black rectangles to redefine the basis vectors of $\boldsymbol{H}$ with magnitude $h$.**

be then computed with:

$$\Sigma_i = \boldsymbol{H}\boldsymbol{D}_i\boldsymbol{H}^\top \quad \text{where} \quad \boldsymbol{D}_i = h_i\boldsymbol{I} \quad \text{and} \quad h_i \sim \mathcal{N}(h, \rho), \tag{22}$$

such that we can perturb $h$ with $\rho > 0$ in order to introduce random variations in the output trajectories. In practice, it is possible to use an arbitrary diagonal matrix for $\boldsymbol{D}_i$, but we observe that our formulation provides sufficient variety of results and reduces the number of open parameters of the system.

It is then easy to edit the semi-tied covariances with an interface in which the user can drag the basis vectors of $\boldsymbol{H}$ and scale the value of $h$ (Fig. 8). Because the cost function used in the optimisation is given by a tradeoff between tracking and control cost, it is possible to keep the maximum displacement $d$ (which determines the control weight) to a fixed value proportional to the workspace area. The user can then define the smoothness of the generated trajectory by manipulating $h$, where an increase in $h$ will produce larger covariances and consequently smoother trajectories.

With this interface, a user can interactively explore different stylisations of a target sequence. While the semi-tied covariances enforce a sense of coordination in the movement, the minimisation of the control cost produces smooth trajectories that evoke a natural drawing movement (Fig. 9). A similar method can also be used to generate different stylisation of an input trace, by setting the target loci in correspondence with curvature extrema along the input. We test this approach with the traces of different letters taken from the *UJI Pen Characters Data Set* [32], and observe that the variation of the semi-tied covariance parameters produces trajectories that resemble the original in structure, but possess clearly different handstyles (Fig. 10). We can see in Fig. 10 that our method also allows to easily add smooth ligatures between a character and the next. To do so we simply consider the respective target sequences as a single one, and then remove targets if the angle formed with the previous target and the next is larger than a user-defined threshold.

## 4.2 Generating Asemic Tags

We have seen how the probabilistic formulation of MPC together with a semi-tied covariance formalism can be used to rapidly explore different stylisations of a letter structure, defined as a coarse sequence of targets. This parsimonious representation can be exploited in combination with procedural generation methods. The user is then left with the simplified task of generating coarse point sequences, while the stylised trajectory evolution is generated by optimal control. Here we demonstrate a simple application, in

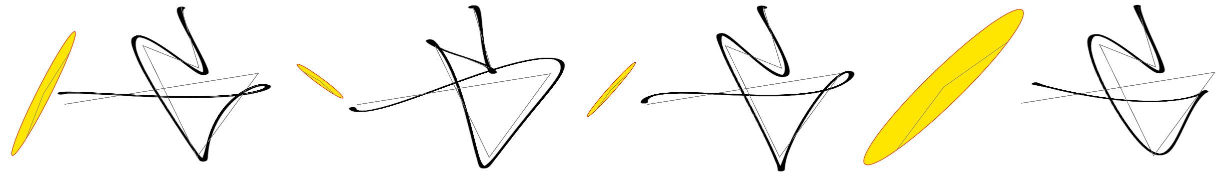

**Figure 9: Handstyles for a letter "G" generated with different kinemic realisations (5th order) of the same target sequence (in gray). In yellow, the respective semi-tied covariances.**

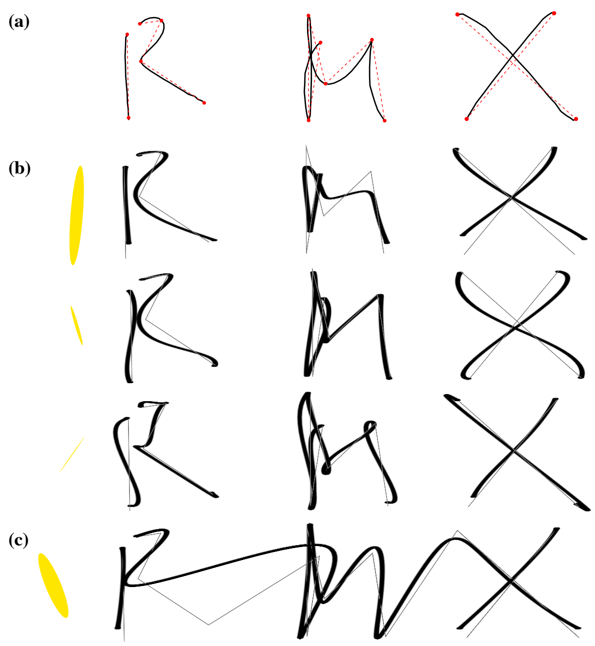


**Figure 10: Handwritten letter stylisation. (a) Traces taken from the UIJI data set and the corresponding target positions (automatically generated). (b) Different stylisations are achieved by using different semi-tied covariance settings. (c) Generation of smooth ligatures. NB: for illustrative purpose we chose a rather low angle threshold of $100°$.**

which glyph-like structures (i.e. asemic letters) are generated using a Genetic Algorithm (GA) [22] and then rendered with different styles by optimal control (Fig. 11.(c,d)).

To define a glyph, we produce a set of $m$ random targets, which are generated in polar coordinates by randomly sampling angles and radii values (Fig. 11.(a)). We then use the GA to determine the ordering of the targets by maximising the distance between consecutive loci (Fig. 11.(b), alike an "inverse" Travelling Salesman Problem (TSP)) and rewarding certain stroke directions that might facilitate motor execution by a drawing hand (*e.g.* down and left-to-right strokes). In the presented examples, we use a GA based upon tournament selection and the PMX crossover operator, which preserve ordering and has previously been used to generate approximate solutions for TSP problems [22].

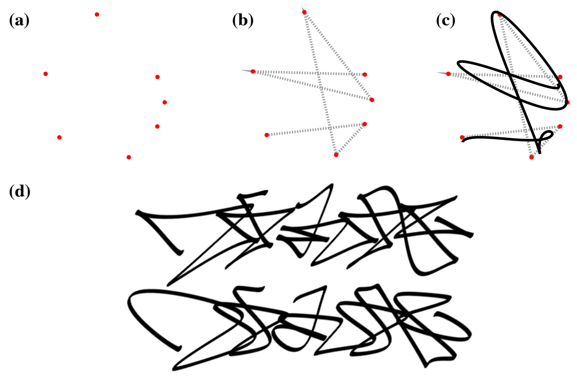


**Figure 11: Asemic tags. Top: three steps of generating a random glyph with a GA and MPC using semi-tied covariances. (d) Five random glyphs are concatenated with ligatures to generate patterns evocative of tags with different handstyles.**

## 5 CONCLUSIONS

We have presented a trajectory and curve generation method based on stochastic optimal control that allows the rapid specification of trajectories that are similar to the ones that can be seen in graffiti handstyles. The trajectories generated by our method reflect physiologically plausible dynamics, which can be exploited to generate realistic animations as well as to drive the smooth motion of a robotic arm [4]. We apply our method for the task of trajectory stylisation with a framework inspired by the work of W.C. Watt [47], in which different stylisations are given by the variations of movement that follow a common geometrical structure. In this paper we explore this problem with an optimal control framework, which allows a user to parametrically generate different hand-styles using a simple user interface and few parameters. In a parallel line of work [3], we are exploring the same type of problem with a data-driven approach, in which stylisations are learned using a type of Recurrent Neural Network (RNN).

The framework described in this paper still has a number of limitations and opens the road for a number of future studies. Currently we limit our examples to planar movements, and do not consider parts of the movement that do not touch the surface (e.g. pen up movements). However, the method we describe is directly applicable to higher dimensions. We plan to explore its direct extension to describe 3D movements as well as additional degrees of freedom in drawing movements, such as pen/brush pressure and orientation.

We have demonstrated how using Gaussians with semi-tied covariances introduces an additional element of coordination across the movement, and results in trajectories that are consistent with different instances of graffiti handstyles. However, this method shows limitations when used with letters such as a rounded "O" (Fig. 12.(a)), which currently require a careful interactive manipulation of each covariance (Fig. 12.(b)), or an adjustment of the maximum displacement parameter $d$.

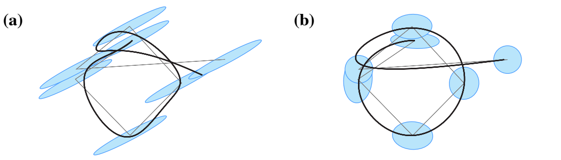


**Figure 12: Difficulties in generating a letter "O". (a) Generated with semi-tied covariances. (b) Generated with user-defined covariances.**

In the presented examples, we have relied on the qualitative evaluation of a number of expert graffiti artists (including the first author) to set the model hyperparameters. We plan to perform more rigorous aesthetic and cognitive-psychological studies in order to evaluate the visual and dynamic quality of the results and to drive future developments of our methods.

*Acknowledgements.* This work has been partly supported by UK's EPSRC Centre for Doctoral Training in Intelligent Games and Game Intelligence (IGGI; grant EP/L015846/1).

## REFERENCES


[1] Z. AlMeraj, B. Wyvill, T. Isenberg, A.A. Gooch, and R. Guy. 2009. Automatically mimicking unique hand-drawn pencil lines. *Computers & Graphics* 33, 4 (2009).

[2] F. Attneave. 1954. Some informational aspects of visual perception. *Psychological review* 61, 3 (1954), 183–93.

[3] D Berio, M Akten, F. Fol Leymarie, M. Grierson, and R. Plamondon. 2017. Calligraphic Stylisation Learning with a Physiologically Plausible Model of Movement and Recurrent Neural Networks. In *Proc. of 4th Int'l Conf. on Movement Computing (MOCO).* London, UK.

[4] D. Berio, S. Calinon, and F. Fol Leymarie. 2016. Learning dynamic graffiti strokes with a compliant robot. In *Proc. of IEEE/RSJ Int'l Conf on Intelligent Robots and Systems (IROS).* Daejeon, Korea, 3981–6.

[5] D Berio, S. Calinon, and F. Fol Leymarie. 2017. Generating Calligraphic Trajectories with Model Predictive Control. In *Proc. of Graphics Interface (GI).* Canada.

[6] D. Berio and F. Fol Leymarie. 2015. Computational Models for the Analysis and Synthesis of Graffiti Tag Strokes. In *Workshop on Computational Aesthetics (CAe),* P. Rosin (Ed.). Eurographics, Istanbul, Turkey, 35–47.

[7] H. Bezine, Adel M. Alimi, and N. Sherkat. 2004. Generation and analysis of handwriting script with the beta-elliptic model. In *Proc. of Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR-9).* IEEE, 515–20.

[8] S. Calinon. 2016. Stochastic learning and control in multiple coordinate systems. In *Intl Workshop on Human-Friendly Robotics.* Genova, Italy, 1–5.

[9] S. Calinon. 2016. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics* 9, 1 (2016), 1–29.

[10] A. d'Avella, P. Saltiel, and E. Bizzi. 2003. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature neuroscience* 6, 3 (2003), 300–308.

[11] E.H. Dooijes. 1983. Analysis of handwriting movements. *Acta Psychologica* 54, 1 (1983), 99–114.

[12] M. Egerstedt and Clyde Martin. 2009. *Control Theoretic Splines: Optimal Control, Statistics, and Path Planning.* Princeton University Press, Princeton Oxford.

[13] J. Feldman and M. Singh. 2005. Information along contours and object boundaries. *Psychological review* 112, 1 (2005), 243.

[14] A. Ferri. 2016. *Teoria del writing, La ricerca dello stile.* Professional Dreamers.

[15] T. Flash. 1983. *Organizing principles underlying the formation of arm trajectories.* Ph.D. Dissertation. Massachusetts Institute of Technology.

[16] T. Flash and A.A. Handzel. 2007. Affine differential geometry analysis of human arm movements. *Biological cybernetics* 96, 6 (2007), 577–601.

[17] T. Flash and B. Hochner. 2005. Motor primitives in vertebrates and invertebrates. *Current opinion in neurobiology* 15, 6 (2005), 660–6.

[18] T. Flash and N. Hogan. 1985. The coordination of arm movements. *Journal of Neuroscience* 5, 7 (1985), 1688–1703.

[19] D. Freedberg and V. Gallese. 2007. Motion, emotion and empathy in esthetic experience. *Trends in cognitive sciences* 11, 5 (2007), 197–203.

[20] F.N. Freeman. 1914. Experimental analysis of the writing movement. *Psychological Monographs: General and Applied* 17, 4 (1914), 1–57.

[21] W.T. Freeman, J.B. Tenenbaum, and E.C. Pasztor. 2003. Learning style translation for the lines of a drawing. *ACM Trans. on Graphics (TOG)* 22, 1 (2003), 33–46.

[22] D.E. Goldberg and R. Lingle, Jr. 1985. Alleles Loci and the TSP. In *1st Intl. Conf. on Genetic Algorithms.* L. Erlbaum Associates Inc., 154–159.

[23] P. Haeberli. 1989. Dynadraw: A dynamic drawing technique. (1989). www.graficaobscura.com/dyna/.

[24] C.M. Harris and D.M. Wolpert. 1998. Signal-dependent noise determines motor planning. *Nature* 394, 6695 (1998), 780–784.

[25] 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.

[26] D.H. House and M. Singh. 2007. Line Drawing as a Dynamic Process. In *Proc. of 15th Pacific Conf. on Comp. Graphics & Applications.* IEEE, Maui, USA, 351–60.

[27] M.I. Jordan and D.M. Wolpert. 1999. Computational Motor Control. In *The Cognitive Neurosciences* (2nd ed.), M. Gazzaniga (Ed.). MIT Press.

[28] J. Kimvall. 2014. *The G-word.* Dokument, Stockholm.

[29] J.E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. 2013. State of the "Art": A Taxonomy of Artistic Stylization Techniques for Images and Video. *IEEE Transactions on Visualization and Computer Graphics* 19, 5 (2013), 866–885.

[30] F. Lacquaniti, C. Terzuolo, and P. Viviani. 1983. The law relating the kinematic and figural aspects of drawing movements. *Acta psychologica* 54, 1 (1983), 115–130.

[31] K. Lang and M. Alexa. 2015. The Markov pen: Online synthesis of free-hand drawing styles. In *Proc. of Workshop on Non-Photorealistic Animation and Rendering (NPAR).* Eurographics, Istanbul, Turkey, 203–15.

[32] D. Llorens Piñana, F. Prat, Marzal Varo, and others. 2008. The UJIpenchars Database: A Pen-Based Database of Isolated Handwritten Characters. (2008).

[33] P. Morasso. 1981. Spatial control of arm movements. *Experimental Brain Research* 42, 2 (1981), 223–7.

[34] P. Morasso. 1986. Understanding Cursive Script as a Trajectory Formation Paradigm. *Graphonomics* 37 (1986), 137–67.

[35] H. Nagasaki. 1989. Asymmetric velocity and acceleration profiles of human arm movements. *Experimental Brain Research* 74, 2 (1989), 319–26.

[36] A. Pignocchi. 2010. How the Intentions of the Draftsman Shape Perception of a Drawing. *Consciousness and Cognition* 19, 4 (2010), 887–898.

[37] R. Plamondon. 1995. A Kinematic Theory of Rapid Human Movements. Part I. *Biological cybernetics* 72, 4 (1995), 295–307.

[38] R. Plamondon, C. O'Reilly, J. Galbally, A. Almaksour, and É. Anquetil. 2014. Recent developments in the study of rapid human movements with the kinematic theory. *Pattern Recognition Letters* 35 (2014), 225–35.

[39] R. Plamondon, C. O'Reilly, C. Remi, and T. Duval. 2013. The Lognormal Handwriter: Learning, Performing and Declining. *Frontiers in Psychology* 4, 945 (2013).

[40] B. Rohrer and N. Hogan. 2003. Avoiding spurious submovement decompositions: a globally optimal algorithm. *Biological cybernetics* 89, 3 (2003), 190–199.

[41] R. Sosnik, B. Hauptmann, A. Karni, and T. Flash. 2004. When practice leads to co-articulation: the evolution of geometrically defined movement primitives. *Experimental Brain Research* 156, 4 (2004), 422–438.

[42] A.K. Tanwani and S. Calinon. 2016. Learning Robot Manipulation Tasks With Task-Parameterized Semitied Hidden Semi-Markov Model. *IEEE Robotics and Automation Letters* 1, 1 (2016), 235–242.

[43] H.L. Teulings and L. Schomaker. 1993. Invariant properties between stroke features in handwriting. *Acta psychologica* 82, 1 (1993), 69–88.

[44] E. Todorov and M.I. Jordan. 2002. Optimal feedback control as a theory of motor coordination. *Nature neuroscience* 5, 11 (2002), 1226–1235.

[45] Y. Uno, M. Kawato, and R. Suzuki. 1989. Formation and control of optimal trajectory in human multijoint arm movement. *Biological cybernetics* 61, 2 (1989), 89–101.

[46] P. Viviani and R. Schneider. 1991. A developmental study of the relationship between geometry and kinematics in drawing movements. *Journal of Experimental Psychology: Human Perception and Performance* 17, 1 (1991), 198–218.

[47] W.C. Watt. 1988. Canons of alphabetic change. In *The Alphabet and the Brain: The Lateralization of Writing,* D. de Kerckhove and C.J. Lumsden (Eds.). Springer, 122–152.

[48] D.M. Wolpert, J. Diedrichsen, and J.R. Flanagan. 2011. Principles of sensorimotor learning. *Nature Reviews Neuroscience* 12 (2011), 739–51.

[49] M. Zeestraten, S. Calinon, and D. G. Caldwell. 2016. Variable Duration Movement Encoding with Minimal Intervention Control. In *Proc. of Int'l Conf. on Robotics and Automation (ICRA).* IEEE, Stockholm, Sweden, 497–503.

# Generating Calligraphic Trajectories with Model Predictive Control


Daniel Berio* Goldsmiths, University of London

Sylvain Calinon† Idiap Research Institute

Frederic Fol Leymarie‡ Goldsmiths, University of London



## Abstract

We describe a methodology for the interactive definition of curves and motion paths using a stochastic formulation of optimal control. We demonstrate how the same optimization framework can be used in different ways to generate curves and traces that are geometrically and dynamically similar to the ones that can be seen in art forms such as *calligraphy* or *graffiti art*. The method provides a probabilistic description of trajectories that can be edited similarly to the control polygon typically used in the popular spline based methods. Furthermore, it also encapsulates movement kinematics, deformations and variability. The user is then provided with a simple interactive interface that can generate multiple movements and traces at once, by visually defining a *distribution* of trajectories rather than a single one. The input to our method is a sparse sequence of targets defined as multivariate Gaussians. The output is a dynamical system generating curves that are natural looking and reflect the kinematics of a movement, similar to that produced by human drawing or writing.



**Index Terms:** Computer Graphics [I.3.3]: Picture/Image Generation—Line and curve generation; Computer Graphics [I.3.6]: Methodology and Techniques—Interaction techniques; Computer Applications [J.5]: Arts and Humanities—Fine arts


## 1 Introduction

The hand drawn curves that can be seen in certain art forms such as calligraphy and graffiti art are often the result of skillful and expressive movements that require years to master [29]. The resulting traces often possess subtle qualitative features which are difficult to reproduce through traditional computer graphics methods such as B-splines or Bézier curves. We adopt the hypothesis that some of the aesthetic qualities observable in hand drawn traces are closely associated to the way such traces were created, and in particular with respect to their dynamic properties (i.e. velocity, acceleration, and associated curving behaviour). Such an hypothesis is commonly held amongst visual artists, and is further supported by academic work in theories of art history [17,34], as well as by studies stemming from the fields of psychology and neuroscience. Such studies indicate that the visual perception of marks made by a drawing hand trigger activity in the motor areas of the brain [18,26], and further induce an approximate mental recovery of the movements and gestures underlying the artistic production [19,32].

In computer graphics applications requiring the simulation of hand drawn traces, it then looks advantageous to follow a *movement centric* approach, in which a shape is defined by the movement underlying its production rather than by an explicit definition of its geometry. We argue that doing so simplifies the process of computationally capturing the inherent qualities of human made traces, such as smoothness and variability, which stem from the properties of the motor system and from the type of movements used when drawing. In this paper, we explore this approach with

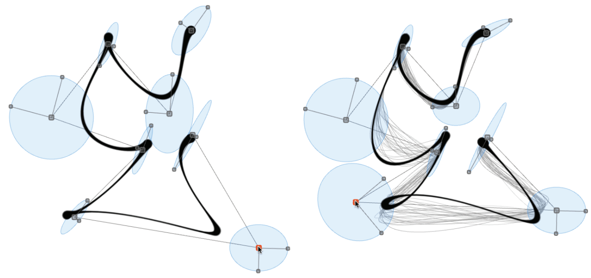

Figure 1: **Left:** Interactively editing of a trajectory through the manipulation of Gaussians. **Right:** interactive editing and visualisation of the trajectory distribution (gray).

a trajectory generation methodology based on *stochastic optimal control*.

Optimal control consists of a family of methods aimed at determining the trajectories of a dynamical system that minimise a given cost or performance criterion [10]. In our study we solve the optimal control problem numerically with a stochastic formulation of *Model Predictive Control* (MPC), a technique that is popular in robotics and industrial related applications. While MPC is typically used in a control system setting, we demonstrate how it can be used as a flexible curve generation tool for computer graphics applications. The control system produces trajectories that have desirable smoothness properties, while sharing common ground with conventional curve generation methods [13].

We formulate the optimisation objective as a discrete sequence of targets that are defined probabilistically in the form of multivariate Gaussian distributions. The centres of each Gaussian define a coarse *control-polygon* which permits interactive manipulation (Fig. 1), behaving similarly to traditional spline/polynomial interpolation methods. In addition, the covariances explicitly describe the *variability* of different parts of the movement, as well as to describe curvilinear features that take into account human movement *coordination* [8]. The optimisation process results in smooth curves and motion paths that are, by design, dynamically and kinematically similar to the ones that would result from a human movement.

The contributions of this paper are two-fold. First, we describe a flexible and interactive trajectory generation method that can be used to synthesise curves for computer graphics applications, while also being able to generate motion paths for computer animation or robotics applications. Second, we show that this methodology is particularly useful for applications requiring the simulation of *human made artistic traces*. We focus on examples involving the production of letter forms as the ones that can be seen in traditional calligraphy as well as contemporary graffiti (street) art. We show that our method is well suited to capture the visual features of such traces, with the additional benefits of facilitating a realistic animation of the trajectory evolution, and providing rich dynamic information that can be exploited to facilitate expressive rendering methods.

The rest of the paper is organised as follows. The next section gives a brief background on related work and further explains the concepts that form the basis for this study. In Sect. 3 we describe the optimisation framework used for the trajectory generation meth-


*e-mail: d.berio@gold.ac.uk

†e-mail: sylvain.calinon@idiap.ch

‡e-mail: ffl@gold.ac.uk

ods, which are then outlined in Sect. 4. These include a stochastic definition of trajectories Sect. 4.1, which can also be reformulated to describe interpolating trajectories (Sect. 4.2), as well as to approximate Bézier curves (Sect. 4.3). Finally, in Sect. 5 we describe how our system can be used in an interactive setting for the purposed goal of generating instances of synthetic calligraphy. Additional mathematical and implementation details are given in the Appendices.

## 2 Related work and Background

While movement synthesis has not been commonly used for the synthesis of artistic imagery (e.g. see [24]), a few examples exist that have proposed approaches that are related to our method. Haeberli implemented DynaDraw, a computer program that allows the user to interactively generate strokes evocative of calligraphy by simulating a mass attached to the mouse position [20]. House et al. [22] as well as Liu et al. [25] generate sketch based renderings by using a *Proportional Integral Derivative* (PID) controller, which controls the evolution of a 2nd order system describing the trajectory of a pen. Shinoda et al. minimize the jerk (time derivative of acceleration) of calligraphic traces defined as B-Splines in order to mimic the effect of running-style Japanese calligraphy [36]. AlMeraj et al. [1], use the minimum-jerk model of human reaching movements [16] to mimic the visual qualities of hand drawn lines. With a motivation similar to ours, Berio and Leymarie [3] apply the Sigma Lognormal handwriting model [33] to interactively generate variations of graffiti like traces. Similarly to our method, letter forms can be defined interactively by defining a control polygon made of a sparse number of targets. In this communication, we propose a method based on stochastic optimal control that, in its different formulations, is applicable to the same type of problem domains as the ones tackled in the previously mentioned studies. The main advantages of our method is that: (i) it encompasses variability and coordination at the descriptive level; and (ii) its generality and flexibility allow a user to experiment with different types of trajectory generation methods and dynamical systems with a single framework.

Typically, hand drawn curves are interactively specified by using a sketch based interface, in which a user traces a curve with a trackpad, mouse or tablet, and the trace is then processed to remove discontinuities and imperfections caused by the digitising device. This process is referred to as "beautification", "neatening" or "fairing", and has been implemented in a number of methods, e.g. [27, 28, 41]. We propose an alternative method to the definition of hand-drawn curves, in which a user defines a coarse series of targets with a point-and-click procedure and a control system defines a trajectory that tracks the targets, resembling traces produced by a human movement.

In this study, we rely on principles that have been observed in the movement science and handwriting analysis/synthesis domains. The velocity profile of rapid and straight reaching motions are characterised by "bell shaped" velocity profiles [30]. Such bell shaped velocity profiles have been modeled with a variety of techniques, including sinusoidal functions [31], Beta functions [5], optimisation methods [16], and lognormals [33]. Many studies propose that smooth arm motions can be described as the space time superposition of a number of "ballistic" movement primitives [40], which are commonly referred to as *strokes*. Each stroke can be represented with the characteristic bell-shaped speed profile. Complex movements tend to show an inverse power relationship between absolute curvature and speed, where curvature extrema usually correspond to minima in the speed profile [44] . Various experiments have exposed the tendency of humans to keep the time of movements relatively independent across different size ranges, aka *isochrony* [44]. Isochrony can be *global*, i.e. for movements and trajectories as a whole, or *local*, for parts of a movement [23]. Complex hand and arm motions tend to be executed in a smooth manner with trajectories that seem to minimise a cost or performance objective [15]. Our method shares relations with a number of models that formulate this objective with the minimisation of the square magnitude of higher order derivatives of position, such as the minimum *jerk* (3rd order) [16], minimum *snap* (4th order) [12] and minimum *crackle* (5th order) [11] models. Humans movements show inherent variability [4], where the variability tends to be higher in parts of a movement that are not critical to the required precision of a task, which is known as the *minimal intervention principle* [42]. Our method is consistent with this principle, and allows a user to explicitly define the required task precision in different parts of a movement through the manipulation of covariances.

We can identify in the literature two dominant representations that are used to describe the spatial evolution of hand-movement paths: *virtual targets* and *via-points*. Virtual targets imply a ballistic stroke representation of movement, and describe the "imaginary" loci at which each stroke is aimed. As a result, these positions do not directly lie along the corresponding motion path. Such a representation has been adopted in a variety of models of handwriting, e.g. [5, 33]. As the name implies, via-points represent landmark positions *along* the trajectory, which are the basis for a number of optimisation-based models of movement (e.g. [12, 16]. In this paper we describe a representation that functions as a "hybrid" between via-points and virtual-targets, in which the evolution of a trajectory is encoded in terms of *multivariate Gaussians*. This hybrid representation is capable of describing ballistic virtual target positions, via-points, as well as more complex spatial constraints, such as forcing a movement to pass through a narrow region of space.

The proposed method also shares relations with interpolating and smoothing splines. Egerstedt and Martin [13] discussed the equivalence between several forms of splines and control theoretic formulations of dynamical systems. The authors cover the case of interpolating splines in which the curve passes through user-defined keypoints, as well as smoothing splines in which a trade-off is found between curve smoothness and curve fitting. It is shown that smoothing splines correspond to the output of a controller found by minimizing quadratic cost functions similar to the ones used in our method. Our method extends this principle to a more generic case, in which each covariance matrix encodes the precision as well as coordination patterns in the movement.

## 3 Optimal control framework

Model Predictive Control (MPC) encompasses a series of numerical methods used to predict the behaviour of a dynamical system, and compute a series of optimal feedback or feedforward commands that will minimise a given cost function and constraints over a given time horizon. In a typical control setting, only the first optimal command is fed to the system, and the optimisation process is repeated iteratively by shifting the time horizon forward to the next time step. As a result, MPC is also commonly referred to as *receding horizon control*. Because our application is focused on curve/trajectory generation rather than control, we perform only a single optimisation step with a time horizon that corresponds to the duration of the trajectory as a whole. From a control perspective, this corresponds to the assumption of a perfect reproduction of trajectory and no external disturbance. As we shall show, this assumption allows us to exploit MPC as a flexible motion and curve synthesis tool.

For the task at hand, we use the simplest form of MPC, such that the objective is unconstrained, the system is linear and the cost function is quadratic. In this case the problem is equivalent to the control problems known as discrete Linear Quadratic Tracking (dLQT). While MPC is based on a well understood control-theoretical background, in our application we focus on a step by step description, which follows and is complemented by mathematical and implementation details in the Appendices. Thanks to the availability of powerful linear algebra libraries and software packages, the method can be implemented in a straightforward manner by following the equations described in the text. For more detailed theory and derivations, the interested reader is referred to [7].

### 3.1 Dynamical system

We model the movement of a pen trajectory by optimising the evolution of a discrete linear time invariant (dLTI) system of order $n$, which results in a discrete sequence of positions $\boldsymbol{x}_t \in \mathbb{R}^D$, where the index $t$ denotes the $t$-th time step. The dynamical system is described with the state space representation

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{A}\boldsymbol{\xi}_t + \boldsymbol{B}\boldsymbol{u}_t, \tag{1}$$

where the system state

$$\boldsymbol{\xi}_t = \left[\boldsymbol{x}_t^\top, \dot{\boldsymbol{x}}_t^\top, \ldots, \overset{(n-2)}{\boldsymbol{x}}{}_t^\top, \overset{(n-1)}{\boldsymbol{x}}{}_t^\top\right]^\top \in \mathbb{R}^{nD}, \tag{2}$$

is given by the position concatenated with its derivatives up to the order $n-1$. The dynamics of the system are determined by the matrices $\boldsymbol{A} \in \mathbb{R}^{nD\times nD}$ and $\boldsymbol{B} \in \mathbb{R}^{nD\times D}$, which fully describe the response of the system to an input command $\boldsymbol{u}_t$. The methods described in this paper function for higher dimensions, but for the scope of this study we focus on planar trajectories, so we assume $D = 2$. While the optimisation framework we will describe can function with arbitrary linear systems, in the examples demonstrated here, we use a chain of $n$ integrators that is controlled by its $n$th order derivative. For example, a system of order 2 will correspond to a system controlled with acceleration commands. The reader is referred to Appendix A for further mathematical details.

### 3.2 Quadratic cost

An optimal trajectory is computed by minimising a quadratic cost function that, for each time step, tries to reduce deviations from a reference state sequence while keeping the amplitude of the control commands low. For a trajectory of $N$ time steps, the cost is given by

$$J = \sum_{t=1}^{N} \left(\hat{\boldsymbol{\xi}}_t - \boldsymbol{\xi}_t\right)^\top \boldsymbol{Q}_t \left(\hat{\boldsymbol{\xi}}_t - \boldsymbol{\xi}_t\right) + \sum_{t=1}^{N-1} \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t, \tag{3}$$

where for each time step, $\hat{\boldsymbol{\xi}}_t$ is the desired state (position and optionally consecutive order derivatives) and $\boldsymbol{Q}_t \in^{nD\times nD}$ and $\boldsymbol{R}_t \in^{D\times D}$ are positive semidefinite *weight* matrices. The weight matrices respectively define the tradeoff between tracking accuracy (*state cost*) and limiting the amplitude of the control commands (*control cost*). In our application, we keep the control cost fixed with a diagonal regularization term $\boldsymbol{R}_t = r\boldsymbol{I}$, where larger values of $r$ produce smoother trajectories.

The optimisation problem can be solved either: (i) in a *batch* form, by solving a large regularized least squares problem; or (ii) *iteratively*, by using dynamic programming and resulting in the a series of time-varying gains. While the latter method is faster, the batch approach directly provides a probabilistic interpretation of the result, which we exploit for stochastic sampling. For implementation details of both methods, the reader is referred to Appendix B.

## 4 Trajectory Generation

In the following paragraphs we describe different trajectory generation methods that can be achieved with MPC. All the proposed methods run at interactive rates and produce trajectories that are smooth and are kinematically similar to the ones that can be seen in movements made by a human when drawing or writing. The control term of the cost function enforces a smooth and continuous trajectory, regardless of continuity of the desired state sequence. As a result, it is possible to define the reference as a sparse sequence of states. This results in a concise and easily manipulable representation of the trajectory geometry that can be exploited in a user interaction scenario.

### 4.1 Minimal intervention trajectories

It has been demonstrated that the combination of MPC with a probabilistic representation can be exploited to capture the variability of multiple human demonstrations in a human-robot interaction scenario [9]. Here, we show how a similar approach can be exploited for the task of trajectory and curve generation.

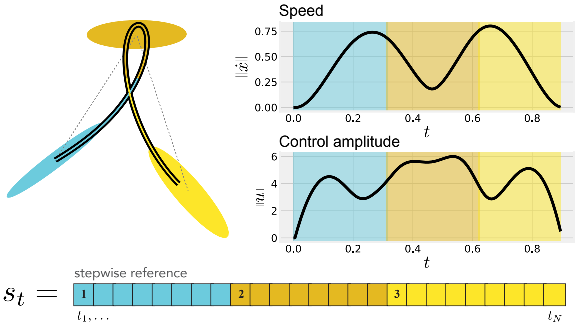


Figure 2: 4th order trajectory generated with Gaussian targets, and the corresponding speed and command (snap) magnitude profiles. Below, schematic visualisation of the state vector for a stepwise reference. The duration of each state is color coded with the corresponding Gaussian.

We describe a trajectory with an ordered sequence of $m$ states, where each state is defined with a multivariate Gaussian distribution $\mathcal{N}_D(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. This representation can be seen as a ballistic decomposition of the movement in $m-1$ strokes, where each stroke $i$ is aimed at the center of the $(i+1)$th Gaussian. With an assumption of perfect *local isochrony* (i.e. a fixed duration for each state), we define a state vector $\boldsymbol{s} \in \mathbb{Z}^N$ in which each consecutive state is indexed $N/m$ times in a stepwise manner (e.g. $\boldsymbol{s} = \{1, 1, 2, 2, 2, 3, \ldots, m\}$, see Fig. 2). The reference state sequence and tracking weights for the whole optimisation horizon are then given by

$$\hat{\boldsymbol{\xi}}_t = \boldsymbol{\mu}_{s_t} \quad \text{and} \quad \boldsymbol{Q}_t = \boldsymbol{C}^\top \boldsymbol{\Sigma}_{s_t}^{-1} \boldsymbol{C}, \tag{4}$$

where the $\boldsymbol{C}$ is the *sensor* matrix, the block entries of which determine what components of the state should be considered in the optimisation objective. For this use case, we only consider the position components, which is described with a sensor matrix

$$\boldsymbol{C} = [\boldsymbol{I}, \boldsymbol{0}, \ldots, \boldsymbol{0}] \in \mathbb{R}^{D\times nD}. \tag{5}$$

This produces zero entries of $\boldsymbol{Q}_t$ for the state derivative terms, which are consequently ignored in the cost function[1]. For the last time step, we set the matrix $\boldsymbol{Q}_N$ identically to Equation 5, but we then add a large constant diagonal value to the derivative terms of matrix that corresponds to a high precision and low variance (we used $1 \times 10^{10}$ in our examples, which for our use case provided consistent results across different system orders without numerical issues). This enforces a zero boundary condition on the state derivatives and brings the movement to a smooth stop.

With this formulation the tracking weights are defined in terms of *required precisions*, and the penalty of deviating from a given state is given by the Mahalanobis distance to the centre of the corresponding Gaussian. The resulting trajectory formation method is therefore consistent with the minimal intervention principle [42]. Each Gaussian functions as a *stochastic target*: as the variance of the Gaussian approaches zero, it increasingly constrains the trajectory to pass

[1] Note that a sensor matrix with all block entries set to $\boldsymbol{I}$ would correspond with an optimisation objective that takes all state derivatives into consideration.

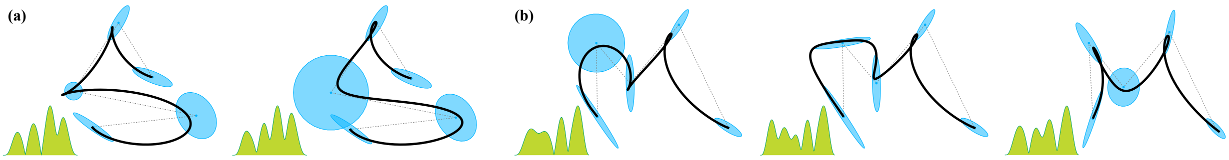


Figure 3: **(a)**, smoothing effect of increasing the variance of a Gaussian. **(b)**, manipulating the trajectory evolution with full covariances. Below each trajectory, its corresponding speed profile.

through its location, effectively behaving like a via-point. A higher variance produces a smoothing effect (similar to smoothing splines), resulting in a behavior that is similar to a virtual target (Fig. 3a). In addition, using full covariances can be exploited to mimic calligraphic effects by allowing the definition of *coordinations* and *directional trends* in the trajectory Fig. 3b).

### 4.1.1 Stochastic sampling

If we consider a Bayesian formulation of the batch version of the minimisation problem, we can interpret the generated trajectory as the center of a *trajectory distribution* [8]. We can then stochastically sample this distribution in order to generate a possibly infinite number of variations over the mean trajectory (Fig. 4, mathematical details in Appendix C). While variations could also be achieved by randomly perturbing the means and covariances of each state, this method allows the user to define the spatial evolution, as well as the variability of the trajectory within a single compact representation. This property is useful for the intended goal of generating calligraphic traces, since it mimics the variability that can be observed in human handwriting, calligraphy and drawing.

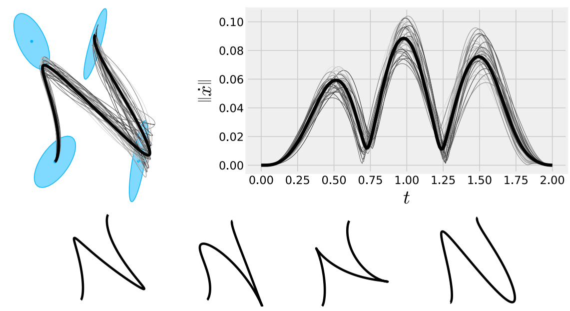


Figure 4: Stochastic sampling of the trajectory distribution for a letter "N". **Top-left:** the mean trajectory (black), random samples from the trajectory distribution (gray) and the corresponding Gaussians. **Top-right:** The corresponding speed profiles. **Bottom:** Selected random samples from the trajectory distribution.

## 4.2 Interpolation with MPC

With a slightly different formulation of the same optimisation framework, we can generate various types of interpolating trajectories. For this task, we define a series of *key points*[2] $\{\boldsymbol{v}_i\}_{i=1}^m$ and the corresponding time steps $\{t_i\}_{i=1}^m$. While in the previously described formulation, we have specified the tracking costs in a stepwise fashion for the purpose of interpolation, we use here a *sparse* reference (Fig. 5, top). Intuitively, this corresponds to an optimisation objective that prioritises trajectory smoothness and only enforces tracking the given states at the given time steps. To achieve this objective,

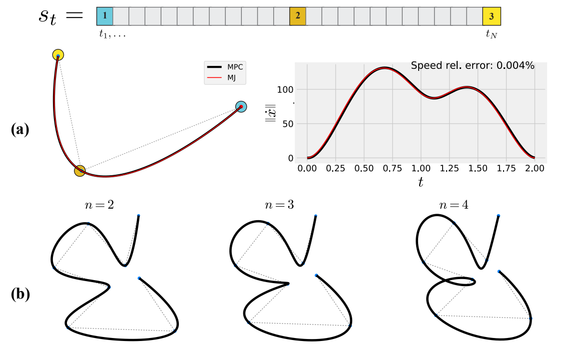


Figure 5: Interpolation with MPC. **a**, comparison of MPC (black) with a closed form solution of a minimum jerk trajectory with 1 via-point (red). On top, a color coded schematic of the corresponding state vector. **b**, examples of interpolating trajectories of increasing order $(2, 3, 4)$.

we set all desired states and weights to zero except for the time occurrence of each key-point with

$$\hat{\boldsymbol{\xi}}_t = \begin{cases} \boldsymbol{C}^\top \boldsymbol{v}_{t_i} & \text{if } t_i = k \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}, \tag{6}$$

and

$$\boldsymbol{Q}_t = \begin{cases} \boldsymbol{C}^\top \boldsymbol{I} \boldsymbol{C}, & \text{if } t_i = k \\ \boldsymbol{I}, & \text{if } t = N \\ \boldsymbol{0}, & \text{otherwise.} \end{cases} \tag{7}$$

This expresses the penalty of deviating from key-point positions at a given time step $k$ with an identity covariance, which reduces to a state cost given by the Euclidean norm to the key-point position. In order to minimize the command amplitude across the whole movement, we then set $\boldsymbol{R}_t = \lambda \boldsymbol{I}$, where $\lambda$ is a very small regularisation term ($1 \times 10^{-15}$ for the examples given in this section). Note that the matrix $\boldsymbol{C}$ is again used to determine the number of derivatives that influence the cost of each via-point. The last entry of $\boldsymbol{Q}_t$ is always set to a full identity matrix, which again enforces a zero boundary condition for the whole state of the system. This formulation allows us to use MPC to produce close numerical approximations of a number of trajectory generation methods, such as polynomial interpolation/splines [14] as well as "minimum square derivative" (MSD) methods such as the minimum jerk model [38]. As a comparative example, we demonstrate that our method can closely approximate curved minimum jerk trajectories with 1 via-point (Fig. 5a), the closed form solution for which is defined by Flash and Hogan in the form of a quintic polynomial [16]. It should be noted that while the MSD methods predict a time difference between consecutive via-points that is *approximately* equal across a movement, the exact time occurrences of each via-point are predicted by the models as part of the optimisation. This will result in the via-point occurring in the proximity of a curvature extrema of the

[2] We adopt the term *key-points* rather than *via-points* here, because the latter term does not describe the initial and final point of the trajectory.

trajectory. In the example shown in Fig. 5a, we compute the exact time occurrence as predicted by the minimum jerk model, which can be done by numerically finding the real roots of a polynomial of degree 9 [16]. For more complex trajectories, we currently assume a uniform spacing in time between via-points (Fig. 5b), leaving the optimization of this timing information for further work.

### 4.3 Mimicking Bézier curves

In the following paragraphs we describe how the same optimisation framework can be used to mimic the shape and behavior of (cubic) Bézier curves. The resulting trajectory/curve generation method provides means of interaction almost identical to its parametric counterpart. At the same time, it provides the flexibility of MPC (such as the ability to easily adjust the trajectory smoothness) and also guarantees trajectory smoothness regardless of the configuration of control points. This is particularly useful for calligraphy generation, where the desired trajectories are per se smooth.

It has been shown that cubic Bézier curves [14] and splines [13] can be interpreted as the trajectories of a 2nd order dynamical system which minimise acceleration commands. Indeed, we can see that with the previously described key-point formulation, it is possible to closely approximate a Bézier curve. This can be done by setting also the first order derivative entry of the sensor matrix $\boldsymbol{C}$ to $\boldsymbol{I}$, and then augmenting the key-points with the derivative computed according to the cubic Bézier formulation (Fig. 6a). This method allows to closely approximate a Bézier curve, and can be used with the same constraints and a higher order systems, which results in a higher degree of continuity (Fig. 6b). However, the method has limitations in the definition of piecewise curves, which for example require setting the same velocity where consecutive curve segments connect.

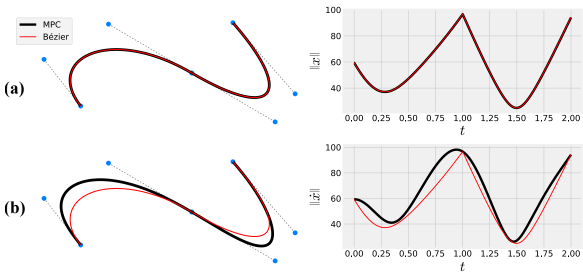


Figure 6: Approximating Bézier curves with a sparse reference and states augmented with velocity. **a**, 2nd order system, which results in an identical curve and speed profile (as shown in [14]). **b**, the same states with a 3rd order system. This produces a slightly different curve with a higher degree of continuity.

We observe that we can also mimic the behavior and shape of a Bézier curve by using a stepwise tracking reference. This can be done by placing isotropic covariance Gaussians centered at each control point of the curve, and then adjusting the influence of intermediate control points on the trajectory by uniformly increasing the variance of each corresponding Gaussian (Fig. 7). The variances are currently set empirically, but we plan to explore methods for automatising this process in further iterations of the study.

At the cost of a less precise approximation, we obtain a curve generation method that produces similar shapes to Bézier curves with a similar representation, and with the additional flexibility of the Gaussian representation and the benefit of always maintaining smooth and physiologically plausible kinematics. The utility of this property in our application is emphasized if we randomly perturb the control points of a letter form and compare the result with the one produced with a Bézier curve (Fig. 8).

While the Bézier curve becomes discontinuous due to the differently oriented tangents at the places where the curve segments

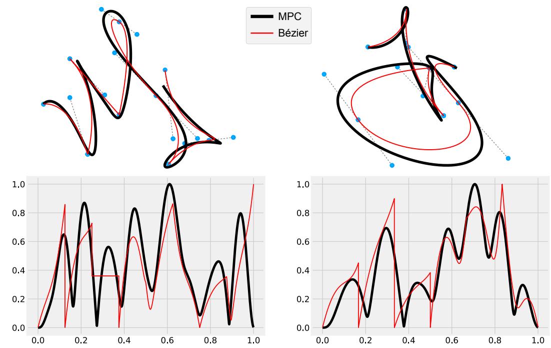


Figure 7: Mimicking Bézier curves (red) with MPC (black) using a stepwise reference, isotropic Gaussians and a $4$th order system. Below, the corresponding speed profiles normalised and superposed for comparison.

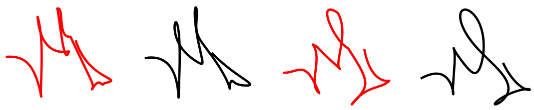

Figure 8: Effect of randomly displacing control point positions with Bézier curves (red) and MPC (black).

meet, the MPC formulation tends to maintain a smooth trajectory regardless of the positions of the control points. This can be exploited as an additional method to generate synthetic variations of a handwriting or calligraphy trajectory, which can be interactively edited with a traditional control point and tangent interface. The same smoothness property can be used to concatenate multiple letter forms with ligatures that evoke a smooth and natural motion, which can be easily achieved by treating the control points of the letters as a single trajectory (Fig. 9).

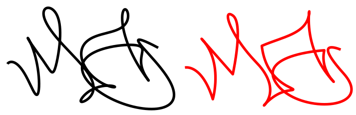

Figure 9: Automatic ligature generation by concatenating the control points of two letters. On the right, a comparative example using Bézier curves.

## 5 Interactive Trajectory Specification and Rendering

The MPC based methods described above are well suited for the interactive definition of trajectories. It is in fact trivial to drag the key-points (Sect. 4.2) or control points (Sect. 4.3) with a typical point-and-click procedure, and it is also simple to interactively manipulate the Gaussians (Sect. 4.1) for the probabilistic case (Fig. 1). Each Gaussian can be edited interactively by manipulating an ellipsoid, where the centre of the ellipsoid defines the mean $\boldsymbol{\mu}_i$, and the axes are used to manipulate the covariance $\boldsymbol{\Sigma}_i$ through its eigendecomposition. The latter can be described with

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Theta}_i \boldsymbol{S}_i^2 \boldsymbol{\Theta}_i^\top , \tag{8}$$

where $\boldsymbol{\Theta}_i$ corresponds to an orthogonal (rotation) matrix, and $\boldsymbol{S}_i$ is a scaling matrix. Here, we describe the 2D case in which the rotation

Figure 10: Top left, a graffiti script (tag) made with a marker by Los Angeles artist “Trixter”. Top right, a user defined “motor plan” for the tag, generated by placing targets near salient positions along the original trace and then adjusting the covariances to follow the original trajectory. Second row, the reconstructed trajectory (left) and one variation made by increasing the regularisation parameter $r$. Bottom row, two random samples from the corresponding trajectory distribution.

and scaling matrices are given by

$$\boldsymbol{\Theta}_i = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \theta = \tan^{-1}\frac{a_2}{a_1}, \quad \boldsymbol{S}_i = \begin{bmatrix} \frac{\|\boldsymbol{a}\|}{2} & 0 \\ 0 & \frac{\|\boldsymbol{b}\|}{2} \end{bmatrix}, \tag{9}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are the major and minor axes of an ellipse, which can be interactively dragged to manipulate the shape of the distribution. Isotropic Gaussians influence the trajectory evolution, in a similar manner to a virtual target or control point, where a small variance will force the trajectory to pass close to the centre. Thinner Gaussians influence the curvilinear evolution of the trajectory, forcing it to follow the direction of the major axis of the ellipse. While the 3D case is currently not implemented, it would be straightforward to adapt the described technique to an arc-ball interface [37], in order to manipulate the 3D rotation components.

In most of our examples we settle with a dynamical of order 4, which we evaluate to give the best balance between trajectory smoothness and a precise control on the trajectory evolution. In order to achieve approximately equal tracking performance across different system orders, we express the control cost in terms of a *maximum displacement* $d$, and compute $R$ based on the low frequency gain of the system with

$$\boldsymbol{R} = \frac{1}{(\omega^n d)^2}\boldsymbol{I} \qquad \text{and} \qquad \omega = 2\pi\frac{T}{m-1}, \tag{10}$$

with $\omega$ empirically chosen, corresponding to the average duration of a stroke. Higher values of $d$ will produce sharper trajectories, while lower values will result in smoother trajectories. On the other hand, because the cost function is defined as a tradeoff between accurate tracking and smooth control, it is possible to keep the value of $d$ fixed depending on the size of the working area, and then interact with the trajectory by only manipulating the Gaussians.

By utilising the stochastic sampling technique described in Sect. 4.1.1, we can interactively visualise the variability of the generated trajectories, while manipulating the Gaussians. This results in a novel form of interactive trajectory editing that, due to its generative nature, directly describes a family of trajectories rather than a single one. As an example, the same interface can be used to rapidly reconstruct, and generate variations of, an existing instance of human made calligraphy (Fig. 10). In this use case, the user first defines a coarse sequence of Gaussians over salient positions along the input trace (approximately in correspondence with curvature extrema), and then adjusts the covariances to modify the trajectory and mimic the curvature and smoothness of the original trace. Variations of the input can then be generated by stochastic sampling.

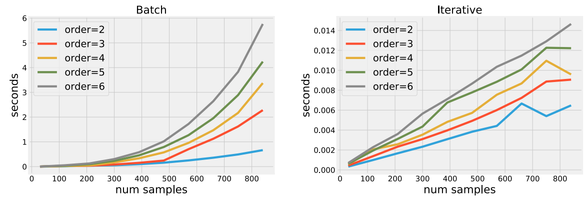


Figure 11: Comparison of performance for the batch and iterative approach.

The proposed methods generate trajectories with smooth derivatives up to the order of the system used in the optimisation. This can be exploited to facilitate painterly and expressive renderings of the trajectory. In this study, we limit ourselves to a simple brush model (see Fig. 1 and 10), which assumes that the amount of paint deposited is inversely proportional to the speed of the pen. To mimic this effect, we sweep a pre-selected texture image along the trajectory with a size that varies as an inverse power function of the instantaneous speed. While this is obviously not an accurate model of a brush or pen, it results in a trajectory rendering that accentuates the perceived dynamics of the trace (see also accompanying video).

Furthermore, the distances between consecutive points along the generated trajectory reflect the smooth kinematics generated by the model. As a result, it is possible to easily generate realistic and natural looking animations, by incrementally sampling points along the trajectory with a fixed time-step. This approach can be used to either generate stroke animations, or to guide the hand motion of a virtual character or robotic arm.

### 5.1 Performance

We have tested our method on a 2,5 GHz Intel Core i7 machine and used OpenGL for hardware accelerated rendering; We have implemented the optimisation code in Python, using the NumPy [43] linear algebra package, as well as in C, using the Armadillo library [35]. Both the batch and the iterative approaches run at interactive rates up to a limit of time steps that depends on the order of the system used in the optimisation. For the examples given in this paper we use an optimisation horizon of approximately 200 time steps, which results in trajectories that are perceived as smooth, and for which the batch and iterative solutions take in average (across orders) 70 ms and 3 ms respectively. The batch approach requires the inversion of a matrix with a dimension which is directly proportional to the number of time steps, and obviously this results in a rapid performance drop as the latter increases (Fig. 11, left). This problem is overcome with the iterative solution, the complexity of which is approximately linear to the number of time steps (Fig. 11, right). On the other hand, the batch solution is more compact and allows the intuitive formulation of probabilistic interpretation of the output and stochastic sampling of the trajectory distribution.

## 6 Conclusion

We have proposed the use of model predictive control (MPC) as a curve generation tool, which can be used in a manner similar to conventional interfaces for polynomial curve generation. Our main contribution is an application to computer graphics of a probabilistic formulation of MPC that explicitly describes intra-movement variability and coordination. We have shown that the same framework can also be used to compute numerical approximations for polynomial interpolation methods such as Bézier curves, or well known trajectory formation methods such as the minimum jerk model. This results in a flexible and general trajectory generation method, that we consider particularly useful for applications that necessitate the simulation of traces such as the ones made by a human when drawing or writing.

While in this paper we focused on the generation of 2D trajectories, the proposed methodology can be generalised to higher dimensions. This opens up the possibility to extend the method to 3D trajectories, as well as taking in consideration the evolution of additional variables, such as the drawing tool orientation or pressure.

In this paper, we have focused on the application of MPC for the generation of movements and traces that mimic the visual qualities of human made graffiti and calligraphy. At this stage, we have relied on the qualitative evaluation by a number of experienced artists and designers ($n = 5$), who have characterised the output of our system as valid instantiations of the targeted hand-styles. In future work, we intend to perform a series of controlled user studies in order to further evaluate the quality of the generated traces and motions. It should be noted that evaluating the aesthetic quality of a visual mark or trace is not a well defined problem, and the notion of "style similarity" can be subjective to the viewer and depends on factors such as cultural and artistic background. On the other hand, we propose, in line with others before us such as Hertzmann [21] or Stacey [39], that the computational study of style is worthwhile, and that a procedure that generates patterns that are perceived as similar to a given artistic style, provides the grounds to establish a potential and computational theory of art that is both *generative* and descriptive [21].

## 7 Acknowledgments

We would like to thank Trixter and David Chang for kindly providing calligraphy samples. This work has been partly supported by UK's EPSRC Centre for Doctoral Training in Intelligent Games and Game Intelligence (IGGI; grant EP/L015846/1).

## A Dynamical system

The continuous-time system matrices for the chain of integrators are given by

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix}, \quad \bar{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix}. \tag{11}$$

The discrete time state matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ used in Equation 1 are then computed using a Zero Order Hold (ZOH) or forward Euler discretisation of $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{A}}$ with a sampling period $\Delta t$, where for the Euler case we simply have

$$\boldsymbol{A} = \Delta t \bar{\boldsymbol{A}} + \boldsymbol{I} \quad \text{and} \quad \boldsymbol{B} = \Delta t \bar{\boldsymbol{B}}. \tag{12}$$

## B Cost function solutions

### B.1 Batch solution

For the batch solution, we express the cost function with

$$J = \left(\hat{\boldsymbol{\xi}} - \boldsymbol{\xi}\right)^{\top} \boldsymbol{Q} \left(\hat{\boldsymbol{\xi}} - \boldsymbol{\xi}\right) + \boldsymbol{u}^{\top} \boldsymbol{R} \boldsymbol{u}, \tag{13}$$

where $\boldsymbol{Q} = \text{blockdiag}\left(\boldsymbol{Q}_1, \boldsymbol{Q}_2, ..., \boldsymbol{Q}_N\right) \in \mathbb{R}^{nDN \times nDN}$, $\boldsymbol{R} = \text{blockdiag}\left(\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_{N-1}\right) \in \mathbb{R}^{DN \times DN}$, and the desired state, current state and control commands are respectively stacked in in large column vectors $\hat{\boldsymbol{\xi}}$, $\boldsymbol{\xi}$ and $\boldsymbol{u}$. We then express all future states as a function of the initial state $\boldsymbol{\xi}_1$, which can be compactly represented in matrix form as

$$\boldsymbol{\xi} = \boldsymbol{S}_{\boldsymbol{\xi}} \boldsymbol{\xi}_1 + \boldsymbol{S}_{\boldsymbol{u}} \boldsymbol{u}, \tag{14}$$

with

$$\boldsymbol{S}_{\boldsymbol{\xi}} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix} \quad \text{and} \quad \boldsymbol{S}_{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix}, \tag{15}$$

where $\boldsymbol{S}_{\boldsymbol{\xi}} \in \mathbb{R}^{nND \times D}$ and $\boldsymbol{S}_{\boldsymbol{u}} \in \mathbb{R}^{nND \times (N-1)D}$. Substituting Equation 14 into Equation 13, differentiating with respect to $\boldsymbol{u}$ and setting to zero results in a regularized least squares estimate of the optimal command sequence, given by

$$\boldsymbol{u} = \left(\boldsymbol{S}_{\boldsymbol{u}}^{\top} \boldsymbol{Q} \boldsymbol{S}_{\boldsymbol{u}} + \boldsymbol{R}\right)^{-1} \boldsymbol{S}_{\boldsymbol{u}}^{\top} \boldsymbol{Q} \left(\hat{\boldsymbol{\xi}} - \boldsymbol{S}_{\boldsymbol{\xi}} \boldsymbol{\xi}_1\right), \tag{16}$$

where the control weight matrix $\boldsymbol{R}$ effectively acts as a Tikhonov regularization term, see e.g. [6]. The command sequence $\boldsymbol{u}$ is then substituted back into (14), resulting in the optimal trajectory $\boldsymbol{\xi}$.

We note that with increasing system orders the amplitude of the commands will grow exponentially, which can result in a badly scaled matrix in the inverse term of Equation 16. To overcome this problem, we first scale the desired states $\hat{\boldsymbol{\xi}}$ by a factor, and the stacked weight matrices $\boldsymbol{Q}$ by the same factor squared. This value is chosen in order to limit the standard deviation of the components of $\hat{\boldsymbol{\xi}}$ below a maximum range. In the examples given in this paper we choose a factor that keeps the standard deviation below $1 \times 10^{-6}$, which gives numerically stable results up to a system order of 5.

### B.2 Iterative solution

A more efficient solution to the optimisation problem can be derived using dynamic programming or an extension of variational calculus known as Pontryagin's Maximum Principle. We refer the interested reader to the work of Bryson [7] for the details of the derivations. It follows that the optimal solution is given in the form of a feedback controller with time varying weighting matrix $\boldsymbol{K}_t$, and the commands are for each time step $t$ given by

$$\boldsymbol{u}_t = \underbrace{-\left(\tilde{\boldsymbol{B}}^{\top} \boldsymbol{P}_t \tilde{\boldsymbol{B}} + \boldsymbol{R}_t\right)^{-1} \tilde{\boldsymbol{B}}^{\top} \boldsymbol{P} \tilde{\boldsymbol{A}}}_{\boldsymbol{K}_t} \tilde{\boldsymbol{\xi}}_t, \tag{17}$$

where

$$\boldsymbol{P}_t = \tilde{\boldsymbol{Q}}_t - \boldsymbol{A}^{\top} \left(\boldsymbol{P}_{t+1} \tilde{\boldsymbol{B}} \left(\tilde{\boldsymbol{B}}^{\top} \boldsymbol{P}_{t+1} \tilde{\boldsymbol{B}} + \boldsymbol{R}_t\right)^{-1} \tilde{\boldsymbol{B}}^{\top} \boldsymbol{P}_{t+1} - \boldsymbol{P}_{t+1}\right) \tilde{\boldsymbol{A}} \tag{18}$$

is a *Riccati difference equation*, which can be solved backwards in time by setting a terminal condition $\boldsymbol{P}_N = \tilde{\boldsymbol{Q}}_N$. In Equation 17 and Equation 18, we introduce the symbols $\tilde{\boldsymbol{\xi}}_t$, $\tilde{\boldsymbol{Q}}_t$, $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$. These respectively denote an *augmented* state vector and tracking weight

$$\tilde{\boldsymbol{\xi}}_t = \left[\hat{\boldsymbol{\xi}}_t^{\top}, 1\right]^{\top} \quad \text{and} \quad \tilde{\boldsymbol{Q}}_t = \begin{bmatrix} \boldsymbol{Q}_t^{-1} + \hat{\boldsymbol{\xi}}_t \hat{\boldsymbol{\xi}}_t^{\top} & \hat{\boldsymbol{\xi}}_t \\ \hat{\boldsymbol{\xi}}_t^{\top} & 1 \end{bmatrix}^{-1}, \tag{19}$$

and augmented system matrices

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix}. \tag{20}$$

This permit us to treat the tracking problem more compactly and efficiently as a *regulation* problem, resulting in a formulation that is equivalent to a Linear Quadratic Regulator (LQR).

## C STOCHASTIC SAMPLING

The optimal trajectory resulting from Equation 14 in the batch solution can be interpreted probabilistically as a trajectory distribution

$$\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\mu_\xi}, \boldsymbol{\Sigma_\xi}) \tag{21}$$

with

$$\boldsymbol{\mu_\xi} = \boldsymbol{S_\xi}\boldsymbol{\xi}_1 + \boldsymbol{S_u}\boldsymbol{u} \quad \text{and} \quad \boldsymbol{\Sigma_\xi} = \sigma \boldsymbol{S_u}\left(\boldsymbol{S_u}^\top \boldsymbol{Q}\boldsymbol{S_u} + \boldsymbol{R}\right)^{-1}\boldsymbol{S_u}^\top, \tag{22}$$

where $\sigma$ is a scaling factor proportional to the mean squared error of the least square estimate in Equation 16, which can be computed automatically in Matlab with the *lscov* command. Additional details on the derivations are given in [8].

This permits the generation of an infinite number of trajectories through stochastic sampling of the distribution, which can be done with the eigendecomposition $\boldsymbol{\Sigma_\xi} = \boldsymbol{V_\xi}\boldsymbol{\Lambda_\xi}\boldsymbol{V_\xi}^\top$, where $\boldsymbol{V_\xi}$ is a matrix of eigenvectors of the symmetric matrix $\boldsymbol{\Sigma_\xi}$, and $\boldsymbol{\Lambda_\xi}$ is a matrix with the respective eigenvalues along the diagonal. We can then stochastically generate variations around the mean trajectory with

$$\boldsymbol{\xi} \sim \boldsymbol{\mu_\xi} + \boldsymbol{V_\xi}\boldsymbol{\Lambda}_{\boldsymbol{\xi}}^{\frac{1}{2}}\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{23}$$

In practice, the covariance matrix $\boldsymbol{\Sigma_\xi}$ will be of high dimension, which will result in slow computation if all eigencomponents are evaluated. It is sufficient here to utilise a reduced subset of eigencomponents with the largest eigenvalues (between 5 and 9 in the provided examples). This can be done at an interactive rate by using the Arnoldi iteration technique [2], which is readily implemented in commonly used linear algebra packages (Matlab®, NumPy in Python, and ARPACK in C).

## REFERENCES


[1] Z. AlMeraj, B. Wyvill, T. Isenberg, A. A. Gooch, and R. Guy. Automatically mimicking unique hand-drawn pencil lines. *Computers & Graphics*, 33(4):496–508, 2009.

[2] W. E. Arnoldi. The principle of min. iterations in the soln. of the matrix eigenvalue problem. *Quat. of App. Maths*, 9(1):17–29, 1951.

[3] D. Berio and F. F. Leymarie. Computational Models for the Analysis and Synthesis of Graffiti Tag Strokes. In P. Rosin, ed., *Computational Aesthetics (CAe)*, pp. 35–47. Eurographics Association, June 2015.

[4] N. A. Bernstein, M. L. Latash, and M. Turvey. *Dexterity and its development*. Taylor & Francis, 1996.

[5] H. Bezine, A. M. Alimi, and N. Sherkat. Generation and analysis of handwriting script with the beta-elliptic model. *Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 8(2):515–20, 2004.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[7] A. E. Bryson. *Dynamic optimization*. Addison Wesley, 1999.

[8] S. Calinon. Stochastic learning and control in multiple coordinate systems. In *Intl Workshop on Human-Friendly Robotics*, 2016.

[9] S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016.

[10] M. J. T. Da Silva. *Pre-computation for controlling character behavior in interactive physical simulations*. PhD thesis, Citeseer, 2010.

[11] J. B. Dingwell, C. D. Mah, and F. A. Mussa-Ivaldi. Experimentally confirmed mathematical model for human control of a non-rigid object. *Journal of Neurophysiology*, 91(3):1158–1170, 2004.

[12] S. Edelman and T. Flash. A model of handwriting. *Biological cybernetics*, 57(1-2):25–36, 1987.

[13] M. Egerstedt and C. Martin. *Control Theoretic Splines: Optimal Control, Statistics, and Path Planning*. Princeton Univ. Press, 2009.

[14] M. B. Egerstedt, C. F. Martin, et al. A note on the connection between Bezier curves and linear optimal control. *IEEE Transactions on Automatic Control*, 49(10):1728–31, 2004.

[15] S. E. Engelbrecht. Minimum principles in motor control. *Journal of Mathematical Psychology*, 45(3):497–542, 2001.

[16] T. Flash and N. Hogan. The coordination of arm movements. *Journal of Neuroscience*, 5(7):1688–1703, 1985.

[17] W. C. Fong. Why Chinese painting is history. *The Art Bulletin*, 85(2):258–80, 2003.

[18] D. Freedberg and V. Gallese. Motion, emotion and empathy in esthetic experience. *Trends in cognitive sciences*, 11(5):197–203, 2007.

[19] J. J. Freyd. Representing the dynamics of a static form. *Memory & cognition*, 11(4):342–346, 1983.

[20] P. Haeberli. Dynadraw: A dynamic drawing technique, 1989. http://www.graficaobscura.com/dyna/.

[21] A. Hertzmann. Non-photorealistic rendering and the science of art. In *8th Int'l Symp. on Non-Photorealistic Animation and Rendering (NPAR)*, pp. 147–57. ACM, 2010.

[22] D. H. House and M. Singh. Line drawing as a dynamic process. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pp. 351–360. IEEE, 2007.

[23] M. I. Jordan and D. M. Wolpert. Computational motor control. In M. Gazzaniga, ed., *The Cognitive Neurosciences*. MIT Press, 2nd ed., 1999.

[24] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Trans. on Vis. & C.G.*, 19(5):866–85, 2013.

[25] J. Liu, H. Fu, and C.-L. Tai. Dynamic sketching: Simulating the process of observational drawing. In *Proceedings of the Workshop on Computational Aesthetics*, pp. 15–22. ACM, 2014.

[26] M. Longcamp, J. L. Anton, M. Roth, and J. L. Velay. Visual Presentation of Single Letters Activates a Premotor Area Involved in Writing. *NeuroImage*, 19(4):1492–1500, 2003.

[27] J. Lu, F. Yu, A. Finkelstein, and S. DiVerdi. Helpinghand: Example-based stroke stylization. *ACM Trans. on Graphics*, 31(4):46, 2012.

[28] J. McCrae and K. Singh. Sketching piecewise clothoid curves. *Computers and Graphics (Pergamon)*, 33(4):452–461, 2009.

[29] C. Mediavilla. *Calligraphy: From Calligraphy to Abstract Painting*. Scirpus, 1996.

[30] P. Morasso. Spatial control of arm movements. *Experimental Brain Research*, 42(2):223–7, 1981.

[31] P. Morasso and F. Mussa Ivaldi. Trajectory formation and handwriting: a computational model. *Biological cybernetics*, 45(2):131–142, 1982.

[32] A. Pignocchi. How the Intentions of the Draftsman Shape Perception of a Drawing. *Consciousness and Cognition*, 19(4):887–898, 2010.

[33] R. Plamondon et al. Recent developments in the study of rapid human movements with the kinematic theory. *Pattern Recognition Letters*, 35:225–35, 2014.

[34] D. Rosand. *Drawing acts: Studies in graphic expression and representation*. Cambridge University Press, 2002.

[35] C. Sanderson and R. Curtin. Armadillo: A template-based C++ library for linear algebra. *Journal of Open Source Software*, 1(2), 2016.

[36] H. Shinoda et al. Generation of cursive characters using minimum jerk model. In *IEEE Proc. SICE*, vol. 1, pp. 730–3, 2003.

[37] K. Shoemake. ARCBALL: A user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface*, pp. 151–6, 1992.

[38] G. Simmons and Y. Demiris. Optimal robot arm control using the minimum variance model. *Journal of Robotic Systems*, 22(11):677–690, 2005.

[39] M. Stacey. Psychological challenges for the analysis of style. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 20(3):167–84, 2006.

[40] H.-L. Teulings and L. Schomaker. Invariant properties between stroke features in handwriting. *Acta psychologica*, 82(1):69–88, 1993.

[41] Y. Thiel, K. Singh, and R. Balakrishnan. Elasticurves: Exploiting stroke dynamics and inertia for the real-time neatening of sketched 2D curves. In *Proc. 24th ACM Symp. on User Interface Software & Technology (UIST)*, pp. 383–92, 2011.

[42] E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5(11):1226–1235, 2002.

[43] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[44] P. Viviani and G. Mccollum. The relation between linear extent and velocity in drawing movements. *Neuroscience*, 10(1):211–8, 1983.