

Tracing curves in the plane: geometric-invariant learning from human demonstrations

Harsha Turlapati¹, Lyudmila Grigoryeva^{2,3}, Juan-Pablo Ortega⁴, Domenico Campolo^{1*}

1 School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore

2 Faculty of Mathematics and Statistics, University of St. Gallen, Switzerland

3 Department of Statistics, University of Warwick, UK

4 School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

*Corresponding author: d.campolo@ntu.edu.sg

Abstract

The empirical laws governing human-curvilinear movements have been studied using various relationships, including minimum jerk, the 2/3 power law, and the piecewise power law. These laws quantify the speed-curvature relationships of human movements during curve tracing using critical speed and curvature as regressors. In this work, we provide a reservoir computing-based framework that can learn and reproduce human-like movements. Specifically, the geometric invariance of the observations, i.e., lateral distance from the closest point on the curve, instantaneous velocity, and curvature, when viewed from the moving frame of reference, are exploited to train the reservoir system. The artificially produced movements are evaluated using the power law to assess whether they are indistinguishable from their human counterparts. The generalisation capabilities of the trained reservoir to curves that have not been used during training are also shown.

1 Introduction

For centuries, human motion has been the focus of studies by scientists and artists. Only in the 19th century, quantitative methods started to emerge [1] which allowed, among other things, formulating *empirical laws* to capture the apparent regularities behind our seemingly effortless motion. As highlighted by Ajemian and Hogan [2], just like what happened in astronomy, progress in motor sciences can be divided into three stages: from “heuristic description”, to “curve fitting”, and to “theoretical synthesis”.

Modern theoretical neuroscience is less than a century old while robotics, another discipline interested in humans, is much younger (in its infancy, one would say, judging from the gap between robots and the simplest forms of animal intelligence). One area where robotics has much to learn from neuroscience is *movement synthesis*.

Perhaps the only genuine similarity between humans and robots of similar size (we are not concerned here with milli-, micro-, or nano-robots) is the fact they both operate in the physical world and are subject to the laws of Mechanics. These laws are very often expressed in geometric forms, e.g. where deviations from “straightness” (e.g. curvature) are related to external forces. These laws are to be obeyed by both humans and robots. Unlike an apple falling from a tree, which can just follow Newton’s laws,

humans (and biological agents, in general) can exert various forms of control, albeit still constrained by the very same laws affecting the apple. For example, this translates into an excess of ‘degrees of freedom’, both kinematic and muscular, which the brain is free to choose from, leading to the concept of *synergies*, a classical problem too vast to review here (the reader is referred to textbooks such as [3], [4]).

While the works above mainly focus on the problem of mapping a redundant kinematic or muscular space onto a task space, typically with much fewer degrees of freedom, the problem addressed in this work can be defined on the task space itself (where the planning is thought to mainly take place [5]) and relates to *curve-tracing tasks*. Since the 80s, studies on curve tracing have focused on both cognitive [6] as well as on motor aspects [7]. In the framework of planar hand movements, where subjects are instructed to freely draw 2D curves, a remarkable regularity has been observed between the velocity v along the curve and the curvature itself κ , which follow the relation

$$v(t) \propto \kappa^{-1/3}(t).$$

For historical reasons, this was named the “2/3 power law” [8] and, in the last four decades, it has been tested, applied and extended under the general name of power laws [9] (the reader is referred to [10] for a recent review). Power laws have an inherently neuro-geometrical flavor [9], linking the given geometry of the problem (i.e. the curve to follow) to a choice made by the brain (i.e. the velocity), a variable by no means prescribed by the given task.

In its basic description, however, the power law states that velocity is proportional to a geometric variable but it does not fix the proportionality constant, which appears to be subject-specific. This is not surprising, as each person is expected to have a preferred speed. The power law then is useful to modulate this preferred speed along a curve. Even for the most recent extensions of the power law [9], there are some parameters that are left undetermined, and that can be thought of as subject-specific. In recent years, power laws have also been shown to be important in robotic applications. For example, during human-machine physical interaction, where nonbiological velocity patterns (i.e. different from a power law) implemented by the robot would result in larger interaction forces [11].

This brings us to the main objective of this work: synthesizing movements that do not only follow an empirical law (power law, in the case of this work) but that have their free parameters adapted to a specific subject. In principle, this can be easily implemented with some form of supervised learning, whereby a demonstration of a human subject tracing a specific curve is used to fit a power law, to be later implemented on a robot. However, a question of generalization would arise next, how to make sure that the power law learned from a specific demonstration (i.e. a specific subject tracing a specific curve) could be ported onto different curves? After all, when given a driving license, we are expected to be able to drive on all sorts of roads, not just the one where the driving test took place.

In this work, we specifically address the two above-mentioned problems of i) learning from a kinematic demonstration of planar curve tracing and ii) generalizing to different (planar) geometries. As mentioned earlier, power laws have an inherently geometric flavor, and the novel aspect of the proposed approach is to formulate the curve-tracing problem in terms of *geometric invariants* and before applying (any) supervised learning methods. Learning geometric invariants, as opposed to rote memorization of a given trajectory, will be shown to be key to generalization.

In the next sections, we shall first recall basic geometric invariants of curves and then present a supervised learning method to fit subject-specific power laws. Finally, we shall show how geometric-invariant laws learned from specific subjects on specific curves can be generalized to different curves. More specifically, in Section 2, we recall that

geometrically invariant quantities such as the curvature enable the reconstruction of a curve parametrically. This motivates our choice to train a learning machine using such invariant quantities, i.e., to learn the geometrical features of the curve itself. Our tool of choice for learning, i.e., the extreme learning machine (ELM), is detailed in Section 3. The setup and preparation of training data, along with the choice of geometric invariant quantities for input and output variables, is also presented. In Section 4, the experimental protocol for the various human subjects in this study and the evaluation criteria for benchmarking the performance of the corresponding trained ELMs with respect to the real human demonstration data is detailed with results. We conclude Section 4 with the discussion of the research gaps in this work and the future scope.

2 Geometric invariants behind curve-tracing

In this section, we shall first recall the geometric invariants for curves in the Euclidean plane. Notation-wise, vectors will be denoted in bold and will be represented as column arrays with components referred to some orthonormal Euclidean frame. The objective of this section is to identify the geometric invariants that help in the analysis of a curve and consequently help choose invariant observations to train a machine learning algorithm that can reproduce human-like movements when tracing similar or different curves.

2.1 Euclidean invariants of 2D curves

Following the convention for differential geometry in [12], consider a 2D curve as a smooth function $\mathbf{c} : \mathbb{R} \rightarrow \mathbb{R}^2$ mapping a scalar parameter $p \mapsto \mathbf{c}(p)$ into the Euclidean plane. The first and second derivatives of $\mathbf{c}$ will be denoted as $\mathbf{c}_p$ and $\mathbf{c}_{pp}$. Consider now a re-parameterization of the same curve $\mathbf{c}(s) := \mathbf{c}(s(p))$. By application of the chain rule, one can write all the higher-order derivatives as :

$$\mathbf{c}_p = \mathbf{c}_4 s_p \quad (1)$$

$$\mathbf{c}_{pp} = \mathbf{c}_{ss} s_p^2 + \mathbf{c}_4 s_{pp}, \quad (2)$$

where s_p and s_{pp} are the first and second derivative of $s(p)$, respectively. Since the re-parametrization s is assumed to be strictly monotonic, the relations (1)-(2) can be inverted as:

$$\mathbf{c}_4 = \mathbf{c}_p / s_p \quad (3)$$

$$\mathbf{c}_{ss} = (\mathbf{c}_{pp} s_p - \mathbf{c}_p s_{pp}) / s_p^3. \quad (4)$$

The *velocity* vector $\mathbf{c}_p$ is by definition tangent to the curve but its magnitude depends on the specific parameterization. By using the re-parameterization $s(p)$ such that $s_p := \|\mathbf{c}_p\| \equiv \sqrt{\langle \mathbf{c}_p, \mathbf{c}_p \rangle}$, where $\langle \mathbf{c}_p, \mathbf{c}_p \rangle := \mathbf{c}_p^T \mathbf{c}_p$ is the *Euclidean inner product*¹, one can guarantee that $\|\mathbf{c}_4\| = 1$, from (3), and define the unitary vector $\mathbf{t}$ *tangent* to the curve as:

$$\mathbf{t} := \mathbf{c}_4 = \mathbf{c}_p / \|\mathbf{c}_p\| \quad (5)$$

Given the tangent, the (unitary) *normal* to the curve $\mathbf{n}$ can be derived via a 90 degree counter-clockwise rotation $\mathbb{J}$ of the tangent vector, that is:

$$\mathbf{n} := \mathbb{J} \mathbf{t}, \quad \text{with} \quad \mathbb{J} := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \quad (6)$$

¹The implicit assumption is that we are representing 2D vectors as column vectors with components relative to some *orthonormal frame*, therefore $\mathbf{c}_p^T \mathbf{c}_p$ (a scalar) corresponds to the sum of the squares of each component.

Albeit not necessarily unitary, another vector ² orthogonal to the tangent is $\mathbf{c}_{ss}$ and its length can be used to define the *Euclidean curvature* μ :

$$\mu := \langle \mathbf{c}_{ss}, \mathbf{n} \rangle = \mathbf{c}_{ss}^T \mathbb{J} \mathbf{t}.$$

So far, all vectors were represented with respect to some (orthonormal) *space* frame. One can consider the family of *moving frames*, for example parameterized by p , with the origin at $\mathbf{c}(p)$, on the curve, and its two axes always aligned with the tangent $\mathbf{t}$ and normal $\mathbf{n}$ vectors. The relationship between spacial and moving frames is captured by the matrix:

$$\mathbf{M} := [\mathbf{t} \quad \mathbf{n}]. \quad (7)$$

It can be easily verified³ that $\mathbf{M}$ is a rotation matrix and in particular that $\mathbf{M}^{-1} \equiv \mathbf{M}^T$. By denoting with a *prime* symbol ($'$) the variable represented with respect to a moving frame, one gets the following identities:

$$\mathbf{t}' := \mathbf{M}^T \mathbf{t} = [1 \quad 0]^T, \quad (8)$$

$$\mathbf{n}' := \mathbf{M}^T \mathbf{n} = [0 \quad 1]^T, \quad (9)$$

$$\mu' := \langle \mathbf{c}'_{ss}, \mathbf{n}' \rangle = \langle \mathbf{c}_{ss}, \mathbf{n} \rangle = \mu. \quad (10)$$

The latter is a particular instance of the invariance of the inner product. The curvature is such an invariant and knowledge of the curvature at any point of a curve is sufficient to reconstruct the entire curve, *modulo a translation and a rotation* [13].

Another important invariant is the *distance* of a given point $\mathbf{r}$ to the curve, see Fig. 1. For any point $\mathbf{c}(p)$ on the curve, one can evaluate the (squared) distance $\|\mathbf{r} - \mathbf{c}(p)\|^2$

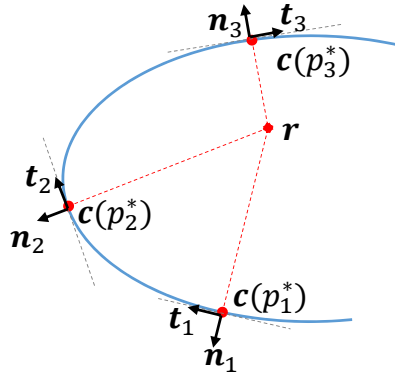


Fig 1. Points on a curve extremizing the distance to a point $\mathbf{r}$. When these points are interior, the tangent at those points is always perpendicular to the segment joining $\mathbf{r}$ to the point itself.

and determine the parameter value p^* as its minimizer within a certain interval $I \subset \mathbb{R}$:

$$p^* := \underset{p \in I}{\operatorname{argmin}} \|\mathbf{r} - \mathbf{c}(p)\|^2. \quad (11)$$

As clarified later, one might not always be interested in a global minimum, and, by properly selecting the interval I , it is possible to exclude unwanted minima. Moreover, whenever p^* in (11) is an interior point, the vector $\mathbf{r} - \mathbf{c}(p^*)$ is normal to the curve $\mathbf{n}$, that is $\langle \mathbf{t}, \mathbf{r} - \mathbf{c}(p^*) \rangle = 0$.

²This can be seen by differentiating $\|\mathbf{c}_4\|^2 = \mathbf{c}_4^T \mathbf{c}_4 = 1$ with respect to s , leading to $\mathbf{c}_{ss}^T \mathbf{c}_4 = 0$.

³By the definition of tangent vector (5), $\langle \mathbf{t}, \mathbf{t} \rangle = 1$ and hence $\langle \mathbf{t}, \mathbf{n} \rangle = \mathbf{t}^T \mathbb{J} \mathbf{t} = 0$ and $\langle \mathbf{n}, \mathbf{n} \rangle = (\mathbb{J} \mathbf{t})^T \mathbb{J} \mathbf{t} = 1$.

One of the main ideas of this paper is to use geometric invariants such as the curvature to train a learning machine. In the specific case of tracing tasks, we expect the machine to learn the intrinsic geometrical features of a curve, independently of possible rotations and translations of the same curve in the workspace.

3 Geometric-invariant learning

In this work, we choose as learning paradigm a class of neural networks called *extreme learning machines* (ELMs) [14] since they have been shown to have the same universality approximation properties as neural networks [15], but they are much easier to train. In contrast with conventional neural networks, where the weights need to be estimated iteratively, the training of ELMs requires only a single linear regression step for the output layer parameters, while the rest of the parameters are randomly initialised. An additional regularization step is also needed to mitigate the potential ill-conditioning of the regression problem. To be more specific, ELMs are defined by the following set of equations:

$$(ELM) \begin{cases} \mathbf{x}_t &= \sigma(\mathbf{c}\mathbf{C}\mathbf{z}_t + \boldsymbol{\zeta}), \\ \mathbf{y}_t &= \mathbf{W}\mathbf{x}_t + \mathbf{b}, \end{cases} \quad (12)$$

where at each sampling time t , the neuronal activity is represented by an internal state vector $\mathbf{x}_t \in \mathbb{R}^{N_x}$ driven by an input vector $\mathbf{z}_t \in \mathbb{R}^{N_z}$. A nonlinear function $\sigma(\cdot)$ is assumed to be the hyperbolic tangent ($\tanh(\cdot)$) that acts elementwise, the parameter $c \in \mathbb{R}^+$ is responsible for scaling of the matrix $\mathbf{C} \in \mathbb{R}^{N_x \times N_z}$ which together with the vector $\boldsymbol{\zeta} \in \mathbb{R}^{N_x}$ is randomly generated. The output of the ELM is represented by the vector $\mathbf{y}_t \in \mathbb{R}^{N_y}$ that is obtained as an affine transformation of the state vector $\mathbf{x}_t$ with $\mathbf{W} \in \mathbb{R}^{N_y \times N_x}$ and $\mathbf{b} \in \mathbb{R}^{N_y}$ the constant matrices.

We emphasize that we shall be using the ELM with the same fixed $\mathbf{C}$ and $\boldsymbol{\zeta}$, in order to enable comparison across the ELMs trained on data collected from different human subjects. More specifically, the randomly generated ELM is kept fixed all along our experiment, and it is only the scaling parameter c , readout matrices $\mathbf{W} \in \mathbb{R}^{N_y \times N_x}$ and $\mathbf{b} \in \mathbb{R}^{N_y}$ as well as the regularization constant λ_{rdg} used in the solution of the regression problem that are tuned for each different subject in the study. This means that the ELM encodes the generic tracing task and the readout parameters account for the specificities and tracing style of the individual subjects.

3.1 ELM setup and data preparation

The starting point in setting up an ELM is choosing the dimension of the input vectors (N_z), the output vectors (N_y) and, more importantly, of the internal state vectors or neurons (N_x). Note that the input and output vectors are determined by the problem itself, i.e., the mapping that needs to be learned. More explicitly, N_z is the number of available explanatory covariates, and N_y is the number of dependent variables. As an example, N_z could be the set of observations about the world and N_y the number of controls, in the case of a robotic task.

For the scenarios of interest in this work, the purpose of the ELM is to learn the unknown dynamics of a given controlled physical process, such as the curve tracing task, based on historical subjects-generated tracing data. In this sense, the most elementary task that one may think of is using as inputs m sequences $\{\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2, \dots, \tilde{\mathbf{z}}_m\}$ containing the positions in time and space of the pencil as driven by the subjects and to train the ELM in order to produce outputs $\{\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \dots, \tilde{\mathbf{y}}_m\}$ associated to each of m samples (or *measurements*) that predict the tracing, that is, $\mathbf{y}_t \equiv \mathbf{z}_{t+1}$.

We shall see later on in Section 3.4 that we might decide to augment the historical input state with additional information, which will, of course, increase the input dimensionality. When it comes to the output vector, apart from predicting the next state given the current input, one might want to try to predict the velocity of the tracing, which also leads to an augmentation of the output dimensionality. In both cases, it is clear that N_y, N_z are specified by the input/output dimensionality of the dynamic process that one is trying to learn.

The dimension of the state space N_x is an architecture hyperparameter that determines the complexity of the model. Even though approximation bounds have been formulated [15] for this machine learning paradigm in which N_x appears as a variable, the lack of knowledge about the stochastic properties of the data-generating process at hand makes it very difficult to use them. In this work and after a trial and error procedure, the number of neurons has been fixed at $N_x = 100$.

3.2 ELM training

Given input and output sequences of historical data, training an ELM consists in determining a matrix $\mathbf{W}$ and a vector $\mathbf{b}$ such that, when the input sequence of historical data is fed into the ELM, the outputs $\mathbf{y}_t$ produced by (12) approximate well the given historical target sequence of outputs. Before training begins, it is useful to prepare the following historical data matrices

$$(training\ set) \begin{cases} \tilde{\mathbf{Z}}_{trn} &:= [\tilde{z}_1 \quad \tilde{z}_2 \quad \dots \quad \tilde{z}_m] \\ \tilde{\mathbf{Y}}_{trn} &:= [\tilde{y}_1 \quad \tilde{y}_2 \quad \dots \quad \tilde{y}_m]. \end{cases} \quad (13)$$

For each $\tilde{\mathbf{z}}_t$, compute $\mathbf{x}_t$ based on the first equation in (12) and construct the matrix of internal states

$$\mathbf{X}_{trn} := [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_m].$$

The ELM training consists in solving the ridge regularized linear regression problem that minimizes the difference between historical output sequence $\tilde{\mathbf{Y}}_{trn}$ and sequence of outputs $\mathbf{Y}_{trn} = \mathbf{W}\mathbf{X}_{trn} + \mathbf{b}$ as predicted by the ELM (12), i.e.

$$(\mathbf{W}^*, \mathbf{b}^*) = \underset{\mathbf{W}, \mathbf{b}}{\operatorname{argmin}} \left\{ \|\tilde{\mathbf{Y}}_{trn} - \mathbf{Y}_{trn}\|_2^2 + \lambda_{rdg} \|\mathbf{W}\|_{\text{Fro}}^2 \right\}.$$

The parameter $\lambda_{rdg} > 0$ determines the strength of the ridge penalty term $\lambda_{rdg} \|\mathbf{W}\|_{\text{Fro}}^2$ that is introduced to solve potential ill-conditioning problems in the covariance matrix of the internal states. The solution to this optimization problem can be written down in a closed form:

$$\mathbf{W}^* = \left(\mathbf{X}_{trn} \mathbf{X}_{trn}^T + \lambda_{rdg} \mathbb{I}_{N_x} \right)^{-1} \mathbf{X}_{trn} \tilde{\mathbf{Y}}_{trn}, \quad (14)$$

$$\mathbf{b}^* = \frac{1}{m} \tilde{\mathbf{Y}}_{trn} \mathbf{1}_m - \frac{1}{m} \mathbf{W}^* \mathbf{X}_{trn} \mathbf{1}_m, \quad (15)$$

where $\mathbf{1}_m := [1, \dots, 1]^T$ is the m -dimensional vector with ones at each entry. The training error or goodness of fit is measured by computing the mean square norm of the difference between $\tilde{\mathbf{Y}}_{trn}$ and $\mathbf{Y}_{trn}$, the latter being computed using the parameters obtained in (14) and (15). The universality properties of ELMs proved in [15] guarantee that this error can be made arbitrarily small by choosing N_x sufficiently large.

3.3 ELM testing and forecasting

A standard problem in machine learning consists in finding the right model complexity (in our case, tuned by the number of neurons N_x) that minimizes the so-called expected

loss or the generalization error. We recall that both too-large and too-small training errors may imply an unsatisfactory generalization error, and hence minimizing the training error for a given sample size does not imply optimal generalization error. We emphasize that we assume no knowledge about the joint distribution of input and target variables and hence do not have access to the generalization error. We hence use a different set of historical data, *testing set*, and use the testing error to assess the performance of the ELM on the learning task of interest.

Specifically, given input and output sequences of s measurements, one forms the following matrices

$$(\text{testing set}) \begin{cases} \tilde{\mathbf{Z}}_{tst} &:= [\tilde{\mathbf{z}}_{m+1} \quad \tilde{\mathbf{z}}_{m+2} \quad \cdots \quad \tilde{\mathbf{z}}_{m+s}] \\ \tilde{\mathbf{Y}}_{tst} &:= [\tilde{\mathbf{y}}_{m+1} \quad \tilde{\mathbf{y}}_{m+2} \quad \cdots \quad \tilde{\mathbf{y}}_{m+s}] \end{cases},$$

where for each $\tilde{\mathbf{z}}_t$ in the testing set ($m < t \leq m + s$), the trained ELM in (12) is used to generate a sequence of predicted outputs $\mathbf{y}_t$, conveniently stored in a matrix

$$\mathbf{Y}_{tst} := [\mathbf{y}_{m+1} \quad \mathbf{y}_{m+2} \quad \cdots \quad \mathbf{y}_{m+s}].$$

The testing error of the ELM on a separate sample than the one used in the training is measured by computing the mean square norm of the difference between $\tilde{\mathbf{Y}}_{tst}$ and $\mathbf{Y}_{tst}$, the latter being computed using the parameters obtained in (14) and (15). Additionally, an optimal ridge regularization constant λ_{rdg} needs to be selected. This is done in practice by reserving a third disjoint sample (we call it *validation sample*) on which we choose the value of λ_{rdg} that minimizes the validation error.

An ELM trained for the one-step ahead prediction task can be *self-looped* to forecast the output in an autonomous manner as in Fig. 2. The testing error of the ELM in this autonomous regime can be also measured.

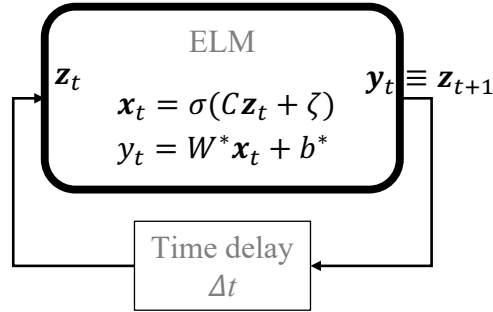


Fig 2. Self-looped configurations of a trained ELM to forecast discrete-time dynamical systems. Note that Δt in a discrete-time experiment corresponds to the sampling rate.

This is the data generation process that will be used in eq. 19 in the next subsection when preparing the training set for learning the curvature-velocity relationship from human tracing data.

3.4 Invariant training of ELMs from 2D curve-tracing data

Consider two curves on the Euclidean plane: the first being a given analytical 2D curve $p \mapsto \mathbf{c}(p)$ parameterized by some (real) parameter p as in Section 2.1 that represents the curve that the subject needs to trace. The second curve is a discrete sequence of points $t_i \mapsto \mathbf{r}(t_i)$ representing the physical motion, with respect to time $t \in \{1, 2, \dots, m\}$ of an agent tasked to trace the given analytical curve. We shall assume that both curves are expressed in coordinates relative to the same orthonormal reference frame and that, as a

result of a measuring process, the physical motion is only available as a dataset consisting of samples of positions $\mathbf{r}_t$ and velocities $\dot{\mathbf{r}}_t$ collected at times $t = 1, 2, \dots, m$.

Training starts with the definition of the training set (13) and, to this end, a complete set of information about the tracing task and the curve would include, at any time $t = 1, 2, \dots, m$, the agent's position $\mathbf{r}_t$ and velocity $\dot{\mathbf{r}}_t$ as well as the closest point on the curve $\mathbf{c}(p^*)$, the tangent $\mathbf{t}$, the normal $\mathbf{n}$ and curvature μ , all evaluated at p^* , where p^* is determined by (11). We notice that in this case the training set depends on the specific reference frame used to define the curve and collect measurements.

In this paper we are interested in defining a *reference-invariant learning* from the available data. Given an ELM, our goal is to design an invariant training procedure that is *independent of any specific choice of reference frame used to define the curve and make measurements*.

In order to achieve this purpose, we propose to represent the available data in terms of *moving frame coordinates* before forming the training set (13). Specifically, at any time $t = 1, 2, \dots, m$, we shall consider the (unique) orthonormal frame centered at $\mathbf{c}(p^*)$ and aligned with $\mathbf{t}$ and $\mathbf{n}$. Algebraically, this corresponds to computing the following *linear* transformations:

$$\mathbf{r}_t \mapsto \mathbf{r}'_t = \mathbf{M}^T(\mathbf{r}_t - \mathbf{c}(p^*)), \quad (16)$$

$$\dot{\mathbf{r}}_t \mapsto \dot{\mathbf{r}}'_t = \mathbf{M}^T \dot{\mathbf{r}}_t, \quad (17)$$

$$\mathbf{c}(p^*) \mapsto \mathbf{c}'(p^*) = \mathbf{M}^T(\mathbf{c}(p^*) - \mathbf{c}(p^*)) \equiv \mathbf{0},$$

$$\mathbf{t} \mapsto \mathbf{t}' \equiv [1 \ 0]^T,$$

$$\mathbf{n} \mapsto \mathbf{n}' \equiv [0 \ 1]^T,$$

$$\mu \mapsto \mu' \equiv \mu,$$

where the transformed (primed) quantities are invariant by definition with an added computational advantage that $\mathbf{c}'(p^*)$, $\mathbf{t}'$, and $\mathbf{n}'$ are *constant* and can hence be removed from the training set. Finally, as remarked earlier in relation to the optimality of p^* in (11), the vector $\mathbf{r}_t - \mathbf{c}(p^*)$ in (16) is necessarily *normal to the curve* when p^* is an interior point. This implies that the first component of $\mathbf{r}'_t$, namely $\langle \mathbf{t}, \mathbf{r}_t - \mathbf{c}(p^*) \rangle$, is always zero while the second component is an *invariant* which can be used to define the task *signed error*

$$e_t := \langle \mathbf{n}, \mathbf{r}_t - \mathbf{c}(p^*) \rangle \equiv \langle \mathbf{n}', \mathbf{r}'_t \rangle = e'_t \quad (18)$$

and which can be used to replace $\mathbf{r}'_t$.

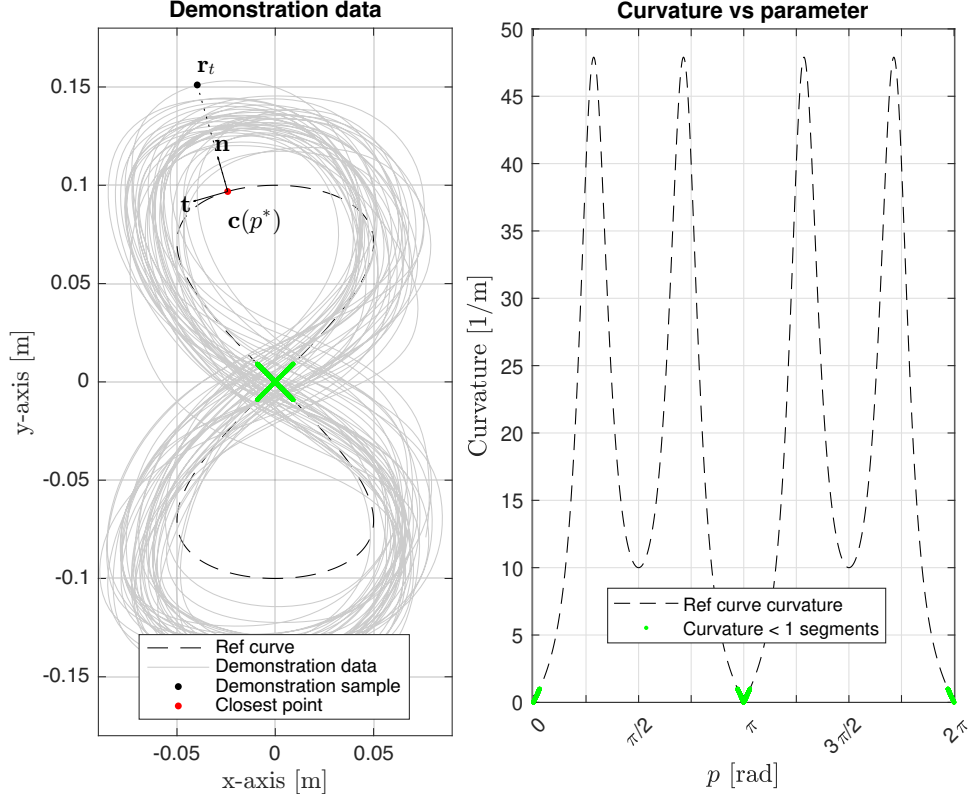
In conclusion, we propose to form the training vectors $\tilde{\mathbf{z}}_t \in \mathbb{R}^3$ in (13) for an ELM based on the following definition:

$$\tilde{\mathbf{z}}_t = \frac{1}{\hat{\sigma}_{\mathbf{z}_{trn}}} (\mathbf{z}_t - \bar{\mathbf{z}}_{trn}), \quad \text{with} \quad \mathbf{z}_t = \begin{bmatrix} e'_t \\ \mu'_t/\mu_0 \\ p_t^*/p_0 \end{bmatrix}, \quad \tilde{\mathbf{y}}_t = \frac{1}{\hat{\sigma}_{\mathbf{y}_{trn}}} (\dot{\mathbf{r}}'_t - \bar{\mathbf{y}}_{trn}), \quad (19)$$

and where at any time $t = 1, 2, \dots, m$, an optimal p^* is evaluated via (11), e'_t is computed as in (18), μ' is computed as (10), $\dot{\mathbf{r}}'_t$ is computed as in (17), while the sample mean $\bar{\mathbf{z}}_{trn}$ and the sample standard deviation $\hat{\sigma}_{\mathbf{z}_{trn}}$ of the training set of observations and, the sample mean $\bar{\mathbf{y}}_{trn}$ and the sample standard deviation $\hat{\sigma}_{\mathbf{y}_{trn}}$ of all training speeds are used to demean and standardize the training dataset. In order to ensure that the inputs to the ELM are within the range of the activation function for each neuron, we use normalisation factors $\mu_0 = 50 \text{ m}^{-1}$ and $p_0 = 10 \text{ rad}$ to enforce that the curvature and the parameter p^* fall in the interval $[-1, 1]$.

4 Learning subject-specific power laws

In this section, the proposed theoretical framework is applied to a set of experimental data. We asked healthy subjects to perform a planar tracing task (shown in Fig. 3a) and, for each subject, we train a randomly initialized ELM.



(a) Subject *S1* tracing data.

(b) Reparameterised curvature function.

Fig 3. (a) The figure of eight was provided as the reference curve to the subjects to perform tracing demonstrations on and the training data for the ELM was prepared by selecting the data-points where the speed was non-zero. (b) Omitted regions with the curvature close to zero are marked in green.

The objective of the study is to assess the similarities between the “motor style” of each human subject and the synthetic motion generated by a trained ELM in self-looped mode as in Fig. 2.

4.1 Human motion and power laws

In this subsection, we discuss the data collection and the power law model which is used to explain the individual motor styles of human subjects. We provide the details regarding the setup of the experiment, data preprocessing, and fitting.

Power law of human motion. In this work we consider a 2-geometry model for evaluating a power law as follows:

$$v = \begin{cases} v_c, & \text{if } \kappa < \kappa_c \\ v_c(\kappa/\kappa_c)^{-\beta}, & \text{otherwise,} \end{cases}$$

where, the *critical* speed and curvature are denoted by v_c and κ_c , respectively, and β is the power law slope. It has been established that humans trace curve segments with low curvature ($\kappa < \kappa_c$) with near constant speeds (v_c) and the power law is thus characterised by these quantities (κ_c and v_c) [9]. To evaluate the power law, we fix $\beta = 1/3$ [9] before obtaining v_c and κ_c through a piecewise logarithmic linear regression on the tracing data using the following model:

$$\log(v) = \log(v_c) + \beta(\log(\kappa_c) - \log(\kappa)) \frac{1 + \text{sign}(\kappa - \kappa_c)}{2}, \quad (20)$$

where κ is evaluated at each t as:

$$\kappa = \frac{|\dot{x}\ddot{y} - \ddot{x}\dot{y}|}{(\dot{x}^2 + \dot{y}^2)^{3/2}} \quad (21)$$

and v is the 2-norm of the vector $[\dot{x}, \dot{y}]^T$ with x and y as in Fig. 3a.

Subjects and protocols. Ten healthy subjects (NTU-IRB Reference: IRB-2018-07-043) were asked to perform a tracing task on a planar robot, called H-Man [16], which is a table-top, portable planar robot. This robot was used only as 2D sensing device, with the motors always off. For the purpose of this study, the participants were instructed to trace out the figure of eight using the passive robot as accurately as possible with the following geometric definition:

$$\mathbf{c}(p) = \begin{bmatrix} 0.05 \sin(2p) \\ 0.1 \sin(p) \end{bmatrix}.$$

Tracing data $\{\mathbf{p}(t)\}$ was collected for individual subjects and down-sampled to a uniform sampling rate of 100 Hz, i.e., $\Delta t = 1/100$ sec. This data was preprocessed and subsequently the piecewise linear power law as in (20) and (21) is fit to the data. Fig. 4 demonstrates the reference geometry, estimated speed, and curvature, along with the tracing data for subject *S1*.

Preprocessing of human tracing data. The tracing data is preprocessed through a set of steps before evaluating the power law and deriving the critical speed v_c and curvature κ_c , which serve to individually identify each subject. First, the scalar speed v and curvature κ are evaluated for each time sample using (21). The tracing data is then additionally filtered to remove the stationary instances by applying a zero-speed threshold. This is also important to avoid evaluating the logarithmic power law when speed or curvature is close to zero. Hence a curvature threshold of 1 m^{-1} is also applied to avoid these instances (see Fig. 3b).

Estimation of human power law. The power law characteristics (the critical speed and curvature) are calculated for each subject's tracing data using the model defined in (20) and labelled κ_c^{real} and v_c^{real} - to attribute to data obtained from *real* human trials. All the codes are implemented in Matlab (2022b), are available on Github, and the computations are run on a MacBook Pro with 2,9 GHz 6-Core Intel Core i9 processor and 32 GB 2400 MHz DDR4 memory.

The data points and the estimated power law characteristics v_c^{real} , κ_c^{real} are shown in Fig. 4(d) for subject *S1* and listed in Table 1 for all the ten subjects in our study.

4.2 Generalization: learning to trace like a human

In this subsection we discuss the implementation of the extreme learning machine (ELM) used to learn the motor style of individual subjects and generalize on the unseen in the training curves.

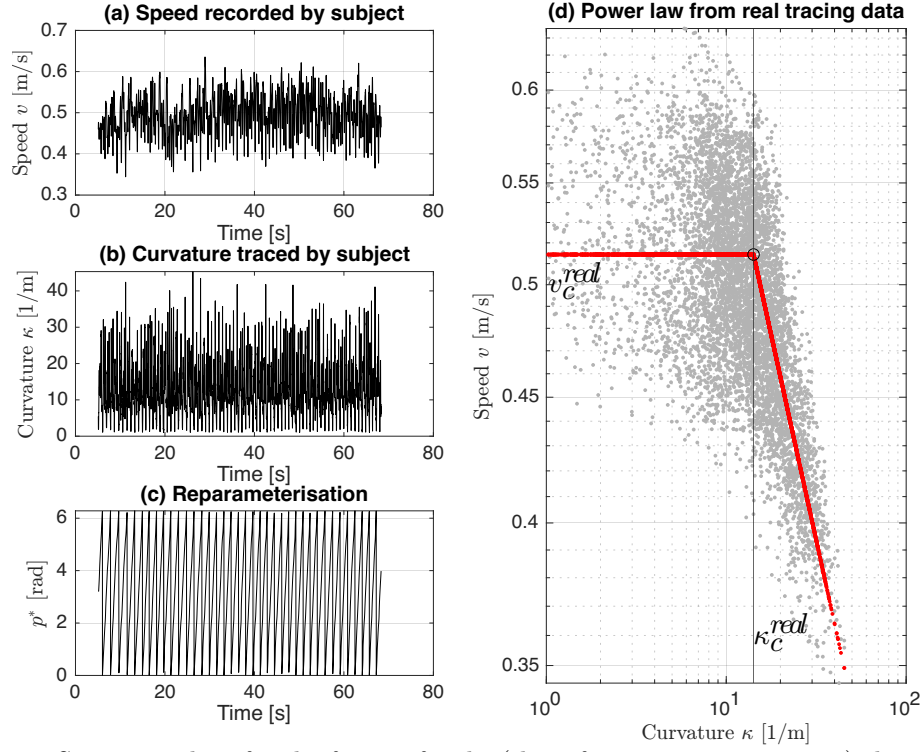


Fig 4. *S1* tracing data for the figure of eight (the reference curve geometry) showing (a) speed v , (b) curvature κ , (c) Euclidean reparameterisation of tracing data $t \mapsto p^*$ to yield the pairs $\{\mathbf{r}_t, \mathbf{c}(p^*)\}$ satisfying (11), and (d) the piecewise linear power law estimated for *S1* tracing data $\{\mathbf{r}_t\}$, yielding critical speed v_c^{real} and curvature κ_c^{real} .

Table 1. Piecewise power law quantities evaluated from tracing data of ten subjects.

	Human power law parameters		ELM power law parameters	
	v_c^{REAL} (m/s)	κ_c^{REAL} (1/m)	v_c^{ELM} (m/s)	κ_c^{ELM} (1/m)
S1	0.514	14.181	0.489	19.001
S2	0.268	15.120	0.245	22.998
S3	0.244	15.525	0.223	22.896
S4	0.148	26.408	0.145	21.699
S5	0.347	16.155	0.333	15.236
S6	0.144	18.891	0.141	16.680
S7	0.136	22.812	0.130	18.487
S8	0.219	19.419	0.201	22.031
S9	0.205	21.123	0.195	22.015
S10	0.094	20.526	0.096	10.742

ELM architecture. We implement the ELM as in (12) with $N_x = 100$ neurons. The bias vector $\boldsymbol{\zeta}$ is chosen to be a zero vector while the connectivity matrix $\mathbf{C}$ is sampled according to the law $\mathcal{U}\{-1, 1\}$.

ELM hyperparameter tuning. Both the scaling parameter $c \in \mathbb{R}^+$ and the regularization $\lambda^{rdg} \in \mathbb{R}^+$ are tuned for each individual subject separately. More explicitly, these hyperparameters are chosen such that they minimize the mean square error of the autonomous $l = 100$ steps operation of the ELM on a set of validation

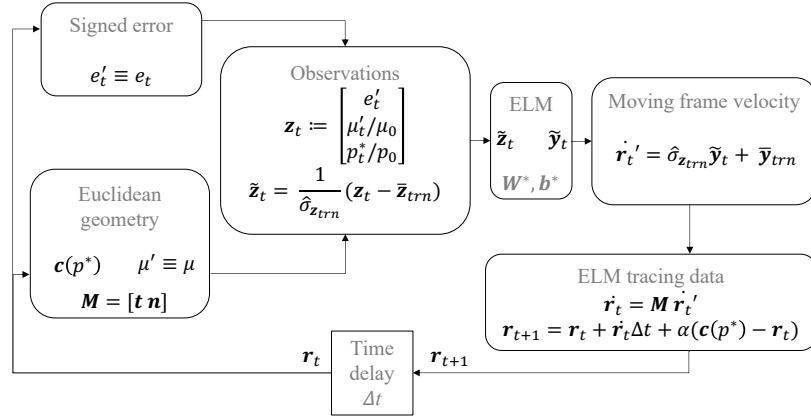


Fig 5. Self-looping ELM: Generalisation of the tracing task is based on the observation set $\mathbf{z}_t$ constituting of the signed error evaluated at the current point with respect to the reference geometry $\mathbf{c}(p^*)$ after the reparameterisation step is done, the Euclidean curvature in moving frame μ' and the parameter p^* - the same structure as the training set data $\{\tilde{\mathbf{z}}_t\}$ (eq. 19). The trained ELM predicts the moving frame velocity $\dot{\mathbf{r}}_t$ that the tracing task requires at time t . The instantaneous spatial frame position of the tracing task is updated to $\mathbf{r}_{t+1}$ with an additional force field $\alpha(\mathbf{c}(p^*) - \mathbf{r}_t)$ imposed to ensure stability, where $\alpha = 0.05$ was heuristically chosen.

samples constructed reserving some part of the total training data. For each subject, we construct a set of indices $\mathcal{S}$ with step 10 starting from 10 and to $\lfloor \frac{1}{4}(m-l) \rfloor$, where m is the length of the total training sample for a given individual subject. For each index $i \in \mathcal{S}$ a particular instance of the i -th training sample with the length $\lfloor \frac{3}{4}(m-l) \rfloor - \frac{1}{2}l$ is used to obtain the estimated readout $\mathbf{W}_i$ and intercept $\mathbf{b}_i$. Next, the associated validation error E_i is measured as the mean squared distance between the result of the autonomous run of the ELM for l steps and the corresponding l -long validation sample. The optimal scaling parameter c and the ridge penalty strength λ^{rg} are obtained as minimizers of the total validation error $\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} E_i$. Once the optimal hyperparameters are found, the total m -long training sample is used to estimate the readout matrix and the intercept for the ELM following the scheme presented in Subsection 3.4.

Data preparation for ELM. For each point in the tracing data $\{\mathbf{r}(t_i)\}_{i=1}^m$, the closest point on the curve in the Euclidean metric was computed using (11) to obtain the corresponding point $p^*(t_i)$. The mapping $t \mapsto p^*(t)$ for the experimental data is shown in Fig. 4. A single point from the tracing data, $\mathbf{r}(t)$, and the corresponding $\mathbf{c}(p^*(t))$ are indicated as an example in Fig. 3. This reparameterisation allows to evaluate moving frame quantities required to define the invariant mapping that will be used to train the ELM.

Remark: It must be noted that the initialisation of the parameter p^* during this step is important and must be done manually to reflect the direction of movement. For instance, for the figure-of-eight curve, the Euclidean reparameterisation at the center of the curve may yield 0 (when the direction of movement is northeastward) or π (when the direction of movement is southeastward). This is a step that requires to be handled manually as the initial value depends on the way the human subject traces the curve, i.e., whether the human tracing is clockwise or counterclockwise. After initialising $p^*(0)$ manually as described, the choice of the interval I in the local minimisation algorithm discussed in (11) is chosen to be $\pm\pi/4$ to limit the locality of search and not produce unwanted minima. This becomes especially important in curves that have overlapping or adjoining loops.

The tracing data of the Extreme Learning Machine. In order to assess the ability of the ELM to learn the motor style of each human subject, we use the same power law model as presented in (20) and (21). After the training of the ELM the self-looping ELM shown in Fig. 5 is initialized at random at an arbitrary point in the vicinity of the reference geometry picked from the training dataset and tasked to self-loop with the criteria defined in Fig. 2 for a duration of 100 seconds, i.e., $100/\Delta t = 10000$ steps. This process was repeated with different initial points for 10 trials to test the robustness of ELM tracing. The tracing performance resulting from this process for the ELM trained on subject *S1* data, is shown in Fig. 6a.

Preprocessing of the ELM tracing data. Similar to human tracing, the dataset is preprocessed through a set of steps before evaluating the power law and deriving the critical speed v_c and curvature κ_c for ELM tracing instances corresponding to individual subjects. Analogously to the human data, the scalar speed v and curvature κ are evaluated for each time sample using (21). A zero-speed threshold and a curvature threshold of 1 m^{-1} are applied to the data. In addition to the steps adopted for the human data, we apply the trimming of the extreme values (outliers) to the ELM autonomous tracing data via keeping only those observations for which the curvature values are within the 85-th percentile.

Estimation of the ELM power law. Parameters of the same model for the power law which is used for human subjects and which is spelled in (20) and (21) are estimated using the preprocessed ELM tracing data. Due to the fact that the piecewise linear regression provides an upward biased estimate of the critical curvature, we modify the estimation procedure to address this issue. More precisely, we add a penalty on the curvature value and choose the penalty strength using cross-validation. For that purpose, we randomly split the ELM tracing data into 80%-long training sample and hold out 20% of the remaining data for validation. We chose the optimal penalty strength as a minimizer of the validation error on the held-out validation set. We label the power law critical curvature and critical velocity characteristics associated with the ELM as κ_c^{ELM} and v_c^{ELM} , respectively. We showcase the results of the power law estimation for subject *S1* in Fig. 6b

Generalization with ELMs for the learned curve. We now illustrate the ability of ELMs to learn subject-specific power laws comparing the power law characteristics v_c and κ_c for the *real* and *ELM* cases. The results of this comparison presented in Table 1. We note that the ELM performance depends on the variability in the intrinsic speed of the subjects. The more regular the training data is (in terms of the speed in a single pass around the curve across multiple loops), the closer the performance of ELM to the power law obtained from real tracing data is.

Generalization with ELMs beyond learned curves. To further test the generalization capabilities of this framework, the ELM was tasked to trace out curves not present in the training data using the self-looping method presented in the next subsection and Fig 7. The figures that were chosen are (i) an affine-transformed figure of eight $\mathbf{c}_1$, (ii) quadrifolium $\mathbf{c}_2$, (iii) ellipse $\mathbf{c}_3$ and (iv) a spiral $\mathbf{c}_4$, to test the ELM's capability of generalising to new curves with a change in the shape but not the curvature range:

$$\begin{aligned} \mathbf{c}_1(p) &= \begin{bmatrix} 2.1213 & -0.70711 \\ 3.5355 & 2.1213 \end{bmatrix} \begin{bmatrix} 0.05 \sin(2p) \\ 0.1 \sin(p) \end{bmatrix}, & \mathbf{c}_2(p) &= \begin{bmatrix} 0.4 \sin(2p) \cos(p) \\ 0.7 \sin(p) \sin(2p) \end{bmatrix}, \\ \mathbf{c}_3(p) &= \begin{bmatrix} 0.6 \sin(p) \\ 0.2 \cos(p) \end{bmatrix}, & \mathbf{c}_4(p) &= 0.05(p + \pi/4) \begin{bmatrix} \cos(5p) \\ \sin(5p) \end{bmatrix}. \end{aligned}$$

The range of the curvature on these new curves is $< 70 \text{ m}^{-1}$. The preprocessing procedure for the ELM data was the same as in the case of the figure of eight

complemented with the additional step of trimming those observations for which the speed values are outside of the 10th and 90th interpercentile range. The power law was estimated using the same estimation technique as for the case of the figure of eight. The tracing performance of the ELM trained using Subject *S1* data for the four new curves is shown in Figs. 7a–7d.

These figures showcase our method’s robustness and generalisation abilities and validate its utility in learning invariant mappings to generalize tracing tasks effectively.

The tracing performance of each self-looping ELM can be characterised by a *synthetic* power law evaluated in the same manner as done for the human tracing data. The next question is whether the *human* power law is comparable to this *synthetic* power law. To answer this question, the parameters that the power law regression yields, i.e., the critical curvature κ_c and critical speed v_c are computed from all the trained self-looping ELMs. The human power law parameters are indicated in red while the corresponding *synthetic* power law parameters are in blue in Fig. 6c. The plot of human-synthetic power law demonstrates that one cannot construct an obvious linear decision boundary that allows to separate the real and ELM instances.

4.3 Discussion

In comparison to other methods for learning tracing tasks in the field of Learning from Demonstrations, our contribution is the geometric nature of learning such tracing tasks. Specifically, one of the applications shown in Dynamic Movement Primitives (DMPs) [17] is the learning of the task of tracing a letter of the alphabet from a demonstration. Their main novelty emphasizes the need for a phase variable that allows for scaling in time, i.e., yielding faster or slower executions of the task. The geometric nature of learning in their work is limited to the specification of a start point and a goal along with a scaling term as a difference between the start and goal. The scaling term carries special significance in ensuring invariance properties when generalising. In other words, without the appropriate scaling proportionate to the change in goal position, the DMP distorts the generalised curve into looking very different. Our approach allows greater geometric generalisation in this sense, because we train on invariant geometric properties of the tracing task, i.e., the signed error, the normalised curvature, and the curve parameter.

Conclusion

This work contributes to the field of *movement synthesis* for robots by proposing a geometric invariant learning framework to perform planar curve tracing tasks.

We first identify that speed and curvature during a tracing task are related through a 2/3 logarithmic empirical law observed in previous studies in neuroscience. We remark that this law characterises *human-like* curve tracing. Building on this we propose to construct an input-output relation between geometric invariants such as the curvature of the curve at the closest point and the instantaneous speed of the human subject and perform supervised learning on this data. Our tool of choice for learning is an ELM which is a reservoir of neurons constructed so that its training is a single linear regression step of the output layer.

Given a human demonstration of the tracing task on a curve, the ability to generalise to a different curve is important to assert the robustness of the learning paradigm. This is shown by the reservoir’s ability to trace out figures like cloverleaves, ellipses, and a spiral.

Acknowledgements

This research is supported by the National Research Foundation, Singapore, under the NRF Medium Sized Centre scheme (CARTIN). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. LG thanks the hospitality of the Division of Mathematical Sciences at NTU, Singapore, which funded the visit that made this collaboration possible.

References

1. Mündermann L, Corazza S, Andriacchi TP. The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications. *Journal of neuroengineering and rehabilitation*. 2006;3(1):1–11.
2. Ajemian R, Hogan N. Experimenting with theoretical motor neuroscience. *Journal of motor behavior*. 2010;42(6):333–342.
3. Rosenbaum DA. *Human motor control*. Academic press; 2009.
4. Shadmehr R, Mussa-Ivaldi S. *Biological learning and control: how the brain builds representations, predicts events, and makes decisions*. Mit Press; 2012.
5. Morasso P. Spatial control of arm movements. *Experimental brain research*. 1981;42(2):223–227.
6. Ullman S. Visual routines. In: *Readings in computer vision*. Elsevier; 1987. p. 298–328.
7. Viviani P, Terzuolo C. Trajectory determines movement dynamics. *Neuroscience*. 1982;7(2):431–437.
8. Lacquaniti F, Terzuolo C, Viviani P. The law relating the kinematic and figural aspects of drawing movements. *Acta psychologica*. 1983;54(1-3):115–130.
9. Bennequin D, Fuchs R, Berthoz A, Flash T. Movement timing and invariance arise from several geometries. *PLoS computational biology*. 2009;5(7):e1000426.
10. Zago M, Matic A, Flash T, Gomez-Marin A, Lacquaniti F. The speed-curvature power law of movements: a reappraisal. *Experimental brain research*. 2018;236(1):69–82.
11. Maurice P, Huber ME, Hogan N, Sternad D. Velocity-curvature patterns limit human–robot physical interaction. *IEEE robotics and automation letters*. 2017;3(1):249–256.
12. Sapiro G. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press; 2001.
13. Gray A. *Modern differential geometry of curves and surfaces with Mathematica*. Boca Raton, FL: Chapman & Hall CRC; 2006.
14. Huang GB, Zhu QY, Siew CK. *Extreme learning machine: Theory and applications*. *Neurocomputing*. 2006;70(1-3):489–501.

15. Gonon L, Grigoryeva L, Ortega JP. Approximation error estimates for random neural networks and reservoir systems. To appear in the *Annals of Applied Probability*. 2022;.
16. Campolo D, Tommasino P, Gamage K, Klein J, Hughes CM, Masia L. H-Man: A planar, H-shape cabled differential robotic manipulandum for experiments on human motor control. *Journal of neuroscience methods*. 2014;235:285–297.
17. Ijspeert AJ, Nakanishi J, Hoffmann H, Pastor P, Schaal S. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation*. 2013;25(2):328–373.

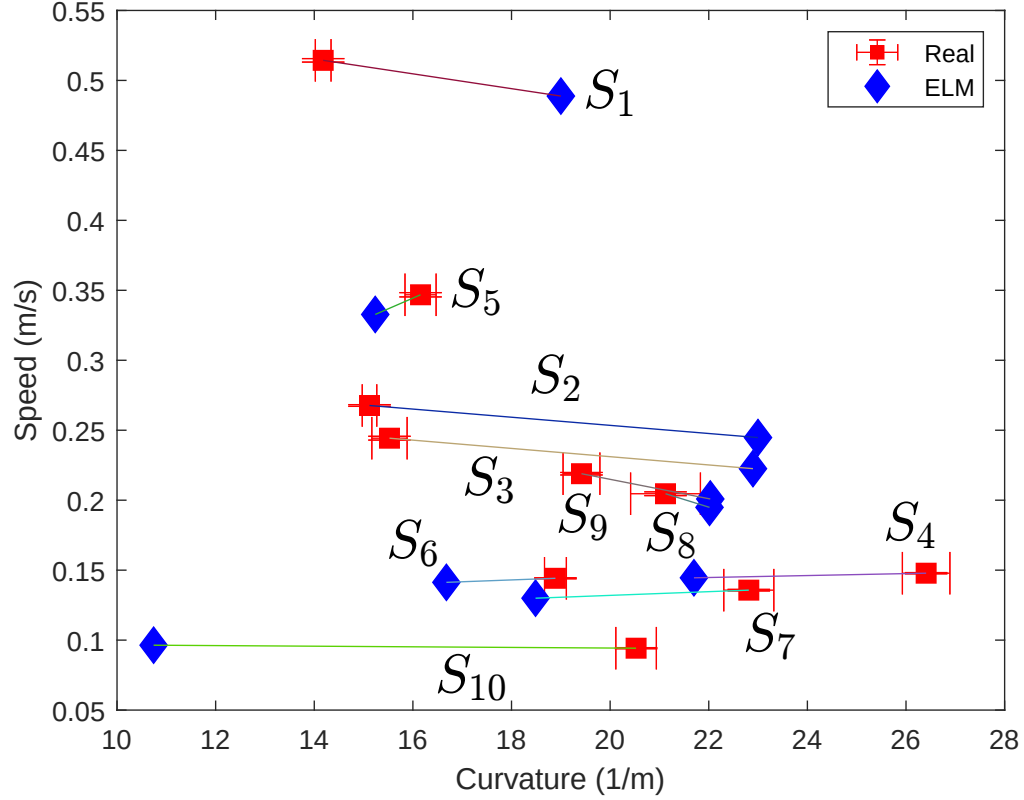
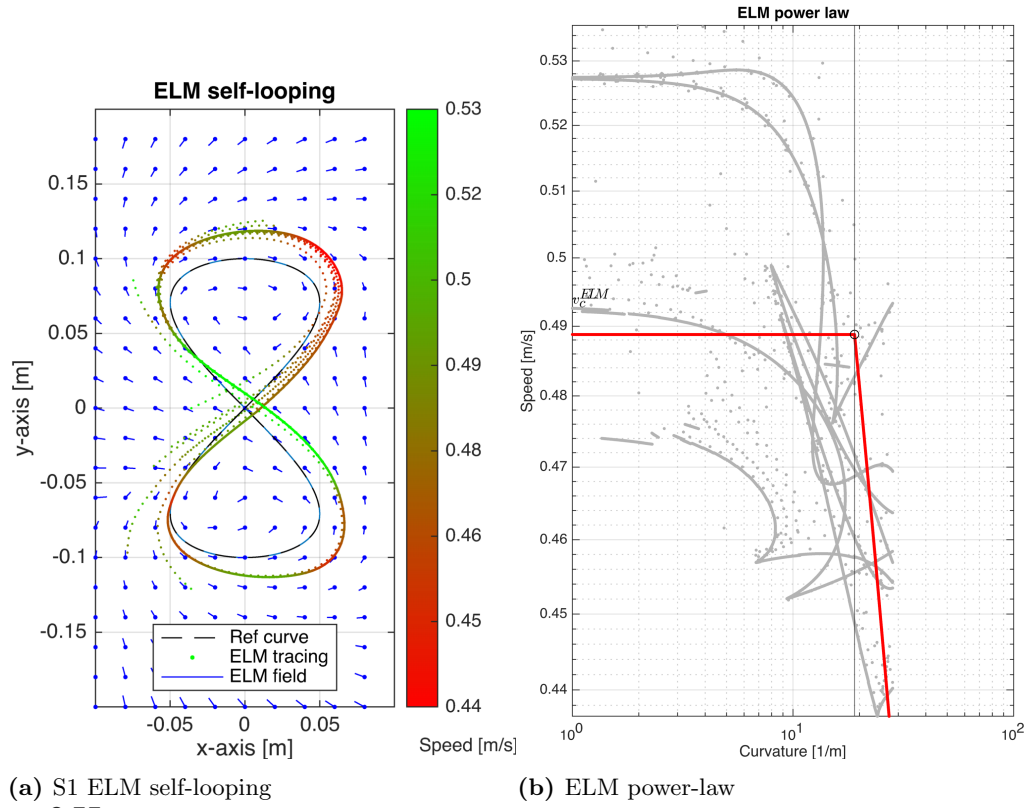


Fig 6. (a) The ELM self-looping behavior may be observed as going slow at the corners and picking up speed along the straighter segments, plotted against the reference curve geometry and the ELM-induced velocity field around the reference geometry. (b) Piece-wise power law evaluated for ELM data. (c) Comparison of the ELM performance with the human demonstration data based on the piece-wise power law.

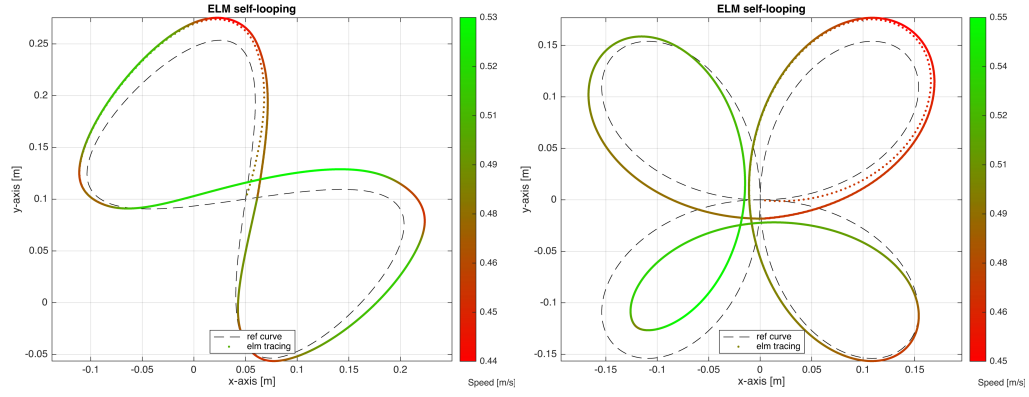
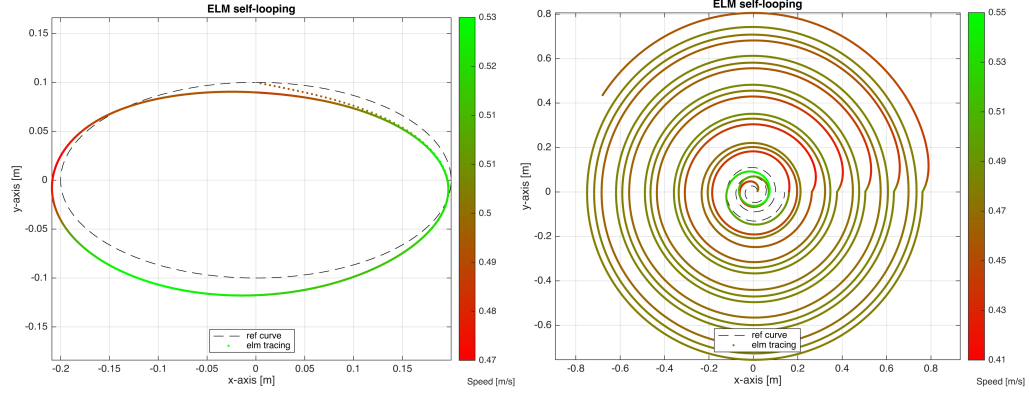
(a) S1 ELM on c_1 .(b) S1 ELM on c_2 .(c) S1 ELM on c_3 .(d) S1 ELM on c_4 .

Fig 7. Tracing performance of ELM trained using demonstration data from S1 on (a) Affine transformed figure of eight, (b) Affine transformed cloverleaves, (c) Ellipse and (d) Spiral.