Generating animal or human locomotion patterns at different speeds, using conceptor morphing

Alina Dima

Jacobs University Bremen
School of Engineering and Science
Campus Ring 1
Bremen, 28759
Germany
August 31, 2015

JACOBS
UNIVERSITY

Generating animal or human locomotion patterns at different speeds, using conceptor morphing

Alina Dima

Supervisor Prof. Dr. Herbert Jaeger

Guided Research in Computer Science

Abstract

Recurrent neural networks can be used to approximate systems that evolve through time arbitrarily well, by receiving input signals and generating output which can be assembled into patterns. A network learns a pattern when it manages to produce it as an output. Although any pattern can be learned provided the appropriate network, learning and later reproducing multiple patterns with the same network is not straightforward, as the network itself changes in the learning process. Conceptors enable top-down manipulation of patterns in a recurrent neural network, enabling incremental learning. The patterns can be organized and further combined to produce new ones, as for instance through morphing. We are aiming at using conceptors to investigate their capability for pattern morph- ing, especially in the case of speed morphing. The ultimate target of the project is to successfully morph human gaits. Applications of pattern morphing can be found in animation and robotics, thus finding better approaches implies better results in motion generation.

Contents

Introduction

1

Recurrent Neural Networks (RNNs) are based loosely on the biological structure of the human brain and consist of directed graphs with weighted connections, being an extension to multi-layer perceptrons. Cycles within the network (known as feedback loops) preserve temporal information, and the network is consequently capable of maintaining an ongoing activation without receiving input. I will refer to the signals fed to the network through training as patterns. RNNs represent nonlinear dynamical systems [FN93] and can be used in tasks such as pattern generation [WS09] or identification and control of dynamic feedback systems [NP90]. Recurrent neural networks have been successfully used in applications in the range of word recognition [Alb+92], signal processing [Kar+92], and pattern recognition [MT91].

Conceptors [Jae14b] [Jae14a] are a fresh addition to the RNN scheme. Each pattern p learned by a network can be attributed to a conceptor C_p, which comes in the form of a linear map. Visually, conceptors embody ellipsoids in the neural space and express how information is stored in the network. After training the network with a desired pattern, its conceptor can be later used to invoke it. This is achieved by inserting the conceptor matrix into the feedback loop, thus feeding $C_p x$ as input to the network, where x is the current state of the network. One can go even further and combine multiple conceptors to obtain a variety of new patterns. The flexible nature of conceptors, coupled with the ability to train the conceptors networks fast using echo state networks [Jae01] makes them promising candidates for pattern morphing.

One of the challenges encountered in machine learning concerns generalization. Two examples of generalization in our context are interpolation and extrapolation; while interpolation seeks to produce new data within the range of the training patterns, extrapolation is the more-challenging attempt to extend this range beyond the trained dataset. Pattern morphing is another instance of generalization, which we will touch on next.

Intuitively, pattern morphing is a combination of multiple signals in order to produce a new signal with a certain degree of similarity to the generating ones. It is often used in robotics as in [IT04]. Speed morphing represents a particular case of pattern morphing, where the goal is to morph two or more signals which exhibit properties that we humans perceive as speed. Let us illustrate speed morphing in human walking gaits; we all know that humans can walk at various speeds, yet often the muscles involved in the different actions are not the same. In this context, training a neural network to produce human walking gaits involves feeding as input information regarding the movements of some representative subset of the joints in the body, and encoding in the output the way a human moves. When patterns of various walking speeds are fed to the network, we would ideally want to produce stable walk-

ing motions of different speeds, either through interpolation or extrapolation. Due to the nature of the task, training a network capable of speed morphing is highly non-trivial.

H. Jaeger has initiated the use of conceptors in pattern morphing [Jae14b]. Promising experiments involving pattern morphing of periodic signal were part of the research. Speed morphing was executed as a form of transition between various patterns in an illustrative video. The current project aims at further investigating the potential of conceptors in pattern morphing, targeted at speed morphing on human walking gaits with the purpose of generating stable, morphed patterns which can be used for an extended period of time rather than brief transitory moments.

Statement and Motivation of Research

2

Training recurrent neural networks poses certain challenges, due to difficulties such as learning long-term dependencies [Ben+94], [Pas+12], bifurcations [Doy93], local minima, slow convergence. A number of strategies have been employed in order to train recurrent neural networks; some techniques are summarized in a tutorial by H. Jaeger in [Jae02]. Backpropagation through time (BPTT)[Wer90], is a direct extension of the classic backpropagation algorithm used for multi layer perceptrons. The cycle-free ordering requirement is satisfied in this approach by unfolding the recurrent network in time. Real-time recurrent learning (RTRL)[WZ89], is a computationally expensive, gradient-descent based method which computes the error stepwise. Echo state networks (ESN) [Jae01], which belong to the greater reservoir computer paradigm, exploit the fact that it suffices to adjust the output weights and ensure that the system can accommodate the desired dynamic. Therefore, training ESNs requires drastically less computation power as the only connections which are learned are the output ones and the network is ensured to be versatile upon initial random value assignment and size.

Switching to a more practical approach, artificial locomotion is needed in disciplines such as robot control or computer animation. Industrial applications of locomotion lie in the area of game industry, film industry, commercial robotics. [WJ12] presents a thorough comparison of techniques and results achieved from more than 50 publications. This paper highlights the fact that despite having made significant progress in generation and control of artificial motion, there are still challenges with stable and efficient motion generation, the area we are concerned about.

Previously conducted research in certain directions in locomotion are relevant for our quest, and represent a starting point. One of the approaches used in locomotion involves central pattern generators (CPGs), neural circuits which produce rhythmic motion that have been observed in biological organisms. Producing human motion in the context of neural networks ultimately means generating periodic patterns, and they have been implemented in a number of ways for producing rhythmic patterns, including echo state networks, as shown in [WS09]. Referring back to pattern morphing, we can see why it is such a challenging task, since the speed often times depends on a multitude of factors acting together, and as such, changing the speed involves more than tuning a single linear parameter. In that regard, frequency modulation is desirable, yet harder to control than other parameters [Wyf+14]. Dynamical approaches have been found to model arbitrary limit-cycle attractors [Aja+13], as they are needed in various locomotion tasks [Jae+12]. Morphing in the context of locomotion has been attempted in various forms in research. [Bos+13] presents a RNN-based model for flexible motor memory in which interpolation has been successful, as well as a limited degree of

extrapolation (coefficients representing the linear combination which generate the morphing go only down to -1).

The goal is to investigate the behaviour of conceptor-based approaches in generalization tasks, as well as the extent to which these methods yield satisfactory results.

Theoretical Setup

3

Neural Networks are fundamental to our experiments. Inspired by the biological structure of the nervous system, various assumptions and training methods have given rise to a multitude of architectures. Recurrent Neural Networks (RNNs) are capable of modelling dynamical systems [FN93], and are the ones used in the experiments.

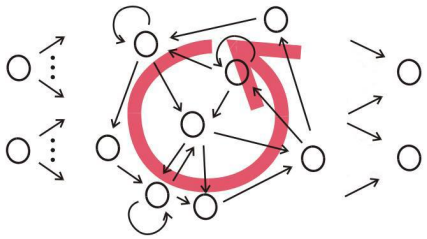


Fig. 3.1: Recurrent neural network. Source [Jae02]

The RNNs which we will be looking at consist of a set of non-spiking neurons, interconnected via unidirectional, weighted synaptic links. Each neuron has an activation, an associated value which is updated discretely, depending on the ingoing connections of its respective neuron. During the update process, the weighted sum of all the ingoing connections is fed to a sigmoid function to limit the range of the result of a predefined interval. A small bias is added to this quantity, and the result is set as the new activation. Intuitively, at any given time, each neuron sends a signal to all of its connected neurons. The signal is proportional to the activation of the sender neuron, and can be dampened depending on the strength of the connection (the weight). In turn, the neurons gather all the signals received and update their activation (see Fig. 3.2).

Apart from the neurons that have been described thus far, which are to be called hidden neurons or internal units, there are two additional types of neurons serving as a mere connection between the network and the outer world: input neurons and output neurons. As one might easily guess, the input neurons transfer the input to the network. The process is similar to the transfer of the activation signal in the discrete time, occurring simultaneously. In this way, a neuron receives at each time step information from both the input units and neighbouring units (I use neighbouring units in the sense of either ingoing or outgoing units, depending on the context). Likewise, the output neurons convert the information back into a readable signal; in our case it will be the signal desired to be learned by the network. In order to be able to pass a broad range of signals, a different transfer function or no function at all may be used.

Due to the complexity of the network, granted by the cyclic pathways which occur, training such a network proves to be an challenging and expensive endeavour. Echo State Networks

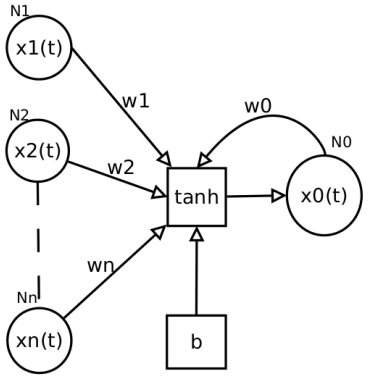


Fig. 3.2: Close-up on information transfer from one state to the next at the neural level. The set $\{N_1, N_2, ...N_n\}$ contains all the units in the network (either internal or input neurons) which have an outgoing connection to N_0. $x_1(t), x_2(t), ...x_n(t)$ represent their state activations; w_i, are the synaptic weights. The activations are multiplied by their respective weights, then summed and fed to the transfer function *tanh*. An additional bias b is added, so that the new activation is calculated as follows: $x_0(t+1) = tanh(\sum_{i=0}^{n} x_i(t)w_i) + b$.

(ESNs) have been designed to deal with the problem by creating a sufficiently rich network, and only training the output weights. A simplified model of ESN without output feedback has been used for the current experiments, in which the information travels from the input units to the hidden layer and ultimately to the output units, just as in Fig. 3.1.

Formally, the transition from a system state to the next, as well as the output generation of the signal, are described by the following update equations:

$$x(n+1) = \tanh(W^* x(n) + W^{in} p(n) + b)$$
$$y(n) = W^{out} x(n),$$

for a network of N hidden neurons, K input neurons and M output neurons, where:

- $p(n)$ represents the K-dimensional input pattern at discrete time n
- $x(n)$ represents the network state at time n, which collects all neural activations of hidden states
- W^{in} is a $N \times K$ matrix which contains the input weights
- W^* is a $N \times N$ weight matrix of hidden units
- $y(n)$ represents a M-dimensional output vector
- W^{out} is a $M \times N$ output weights matrix
- b is the bias of the neuron

From a more abstract point of view, the internal units form a directed acyclic graph with weighted vertices and edges. The internal units can be seen as a function from a vector to

the graph's weights $f : \mathbb{R}^K \to \mathbb{R}^N$; similarly, the output units are represented by a function $f : \mathbb{R}^N \to \mathbb{R}^M$.

3.1 Conceptors

Consider a network of N hidden units. During the learning process of a pattern p, the reservoir becomes excited and incorporates the input signal in its activations $x(n)$. The resulting sequence of network states $x(1), x(2), ...$ represents a point-cloud in the N-dimensional network state space. Using Principal Component Analysis(PCA), the geometry of the point-cloud can be characterized by an ellipsoid, shaped by uncorrelated axes of the point-cloud corresponding to maximum variance. The lengths of these axes represent the singular values of the correlation matrix R of the points $x(1), x(2), ...$

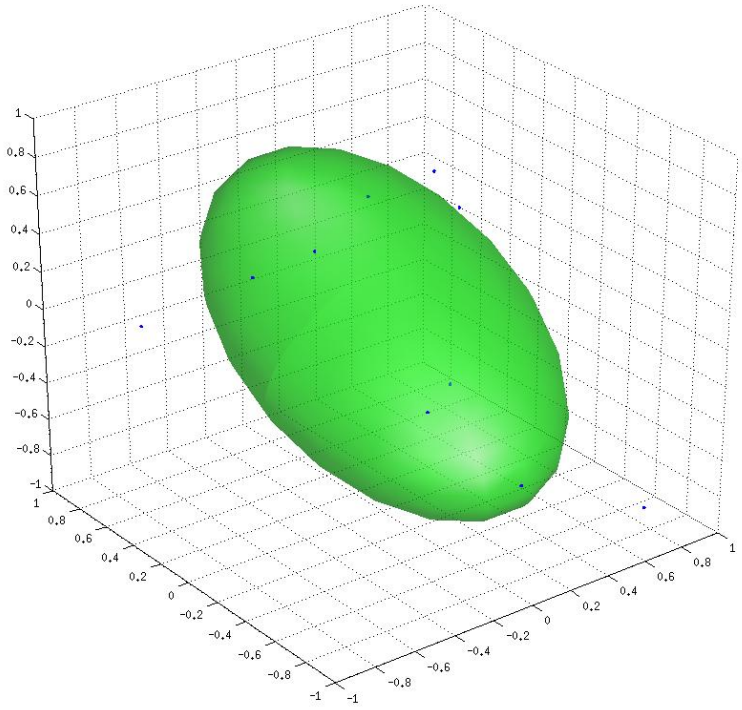


Fig. 3.3: Point cloud with its corresponding ellipsoid generated by principal component analysis. The blue dots mark the data points.

In the next step the ellipsoid is normalized to fit in the N-dimensional unit sphere. A conceptor matrix C is defined as:

$$C = R\left(R + \frac{1}{\alpha^2} I\right)^{-1} \tag{3.1}$$

Thus, singular values s_i of C and the singular values σ_i of R satisfy the equation:

$$s_i = \frac{\sigma_i}{\sigma_i + \alpha^{-2}}, \tag{3.2}$$

, with the mention that now all $s_i \leq 1$.

The parameter α is called an aperture, and it strikes a balance between the correlation matrix R and the identity matrix I.

Alternatively, a conceptor can be seen as a matrix which minimizes the cost function:

$$\sum_{n=1}^{L} \frac{1}{L}||x(n) - Cx(n)||^2 + \frac{1}{\alpha^2}||C||^2 \tag{3.3}$$

The aperture has to be adapted properly for accurate pattern regeneration, as part of the reservoir set-up. A small aperture can cause insensitivity to patterns, while a large one makes the network overexcited and sensitive to noise.

3.2 Learning and Loading a Pattern

The training process of the network consists in computing W^{out}, as well as the conceptor matrices corresponding to each pattern present in training.

At the end of the learning phase, the network should be able to reproduce the input pattern:

$$y(n) = W^{out}x(n) \simeq p(n), \forall n \tag{3.4}$$

We have already established that the conceptor matrix C_j corresponding to pattern p_j is to be computed by the formula (3.1), where the correlation matrix R depends on the network activation states during the learning iterations: $R = (X_j\, X_j^T)/L$, with L being the length of the pattern (or of the relevant bit thereof).

Additionally, in a process called pattern loading, a new matrix W is computed such that it satisfies:

$$Wx(n) \simeq W^*x(n) + W^{in}p_i(n), \tag{3.5}$$

for all training time steps n and patterns p_j (see section 4.1 for details). Following this step, we would expect the following to happen:

$$x(n+1) = \tanh\left(Wx(n)\right), \tag{3.6}$$

which is the case when a single pattern has been learned. However, if multiple patterns are stored, the network cannot decide which pattern to load in the absence of any further information. However, with the use of conceptors any of the learned patterns can be outputted:

$$x(n+1) = C_i \tanh\left(Wx(n)\right), \tag{3.7}$$

3.3 Morphing Patterns

Morphing patterns with conceptors can be done using conceptor matrices. Suppose a reservoir has been trained with patterns p_i and p_j, having corresponding conceptors C_i and C_j. A morphed pattern between p_i and p_j is defined by the following state update equation:

$$x(n+1) = ((1-\mu)C_i + \mu C_j)\tanh(Wx(n)), \tag{3.8}$$

where μ determines how similar the morphed pattern is to any of the initial ones.
For $0 < \mu < 1$, the process is called interpolation between the two patterns, while outside of the range we are dealing with extrapolation. The equation can be easily generalized to multiple patterns.

3.4 Dynamic pattern generation and challenges in pattern morphing

In order to have a better understanding of the patterns we will be working with, as well as the challenges that pattern morphing poses under these conditions, we will briefly introduce some concepts in dynamics. I will follow the introduction found in [ST93].

A dynamical system is a system of equations stipulating the temporal evolution of a point x. A phase space of a complex system is an n-dimensional Euclidian space in which all possible states of the system are represented.

When the time is continuous, the dynamics of a system can be typically defined as a set of first-order ordinary differential equations (ODEs):

$$\dot{x} = F(x), \tag{3.9}$$

where $\dot{x}$ is the derivative of x with respect to time. A flow arises naturally within this context, as a group action of the real numbers onto the n-dimensional space. The flow adds time to the n-dimensional value space, incorporating the evolutions of points in time in the dynamical system. Examples of such dynamical systems are Lorenz and Rössler attractors, which are discussed in more detail in section 4.1.

When the time is discrete and assumes integer values, its dynamics are described by a group action of the integers over the n-dimensional space:

$$x_{t+1} = F(x_t) \tag{3.10}$$

Transitioning from a continuous-time dynamical system to a discrete-time system is realizable using time-1 maps. Since a discrete dynamical system is made of time-cuts in the flow of the underlying continuous dynamical system, transitioning from time t to time $t+1$

is equivalent to finding a map between the two cuts taken at a time difference of 1. This method was used in our experiments to generate rich dynamic patterns starting from ODEs.

An attractor is a subset of the phase space to which initial conditions converge asymptotically as time $t \to \infty$. If that region is a point, it is called a fixed point attractor. For example, the system described by $x \to x^2$ has a fixed point attractor at 0, and a repeller fixed point at 1. Similarly, a repeller is a region of the phase space from which initial conditions diverge as time $t \to \infty$.

Zooming out, we can define families of dynamical systems depending on certain parameters. For instance, the dynamical system $x \to x^2$ is the system resulting when $c = 0$ in the family $x \to x^2 + c$, described by the single parameter c. The typical behaviour is that when the parameters vary just slightly, the resulting dynamical systems are closely related, presenting nearby fixed points, attractors, and repellers. However, it can happen that the qualitative aspect of the system changes drastically, following a small change in parameters, in which case we call it a bifurcation.

An example of bifurcation occurs between walking and running in human gaits. As the succession of steps becomes faster and faster, the running process is marked by a sudden jump. In fact, walking requires at least a leg to touch the ground at all times, while running does not pose the restriction, having periods when no leg touches the ground. Even the sets of muscles used in the two motions differ. Under these circumstances, a continuous transition from walking to running is impossible to achieve.

Bifurcations are true villains in morphing. Whenever they are present, no meaningful morphing can be achieved, since the dynamical properties of the environment differ.

Experimental Setup

4

In this chapter the concrete details regarding experiments set-ups are provided, as well as tools used to measure the outcomes of the experiments.

The experiments are divided into two sections: learning two similar attractor patterns, generating sufficient intermediate morphed signals and analysing the results in 4.1, and speed morphing between walking gaits from motion capture data in 4.3.

Normalized Root Mean Squared Error (NRMSE) is introduced in 4.1.1 as a measure for the similarity between two vectors, and is used to determine how well a pattern has been learned by the network, as well as assess other tools. In 4.2, a way to quantify morphing is discussed.

4.1 Learning attractors

The first part of the experiments consists in learning two versions of the same attractor, where the chances of occurring bifurcations are rather low, in order to measure how well-behaved the morphing process is on dynamical systems of such complexity.

A network of 500 neurons has been used, with two additional input units and two output units. Thus, two dimensions of an attractor could be learned and reproduced with sufficient richness.

Recall the two state update equations:

$$x(n+1) = \tanh(W^* x(n) + W^{in} p(n) + b) \tag{4.1}$$

$$y(n) = W^{out} x(n), \tag{4.2}$$

W^{in} values have been sampled from a normal distribution, and scaled by a factor of set factor called W_{in} scaling.
The bias b has been sampled from a normal distribution, and rescaled by a bias scaling factor.
W has been sampled from a normal distribution and scaled to reach a desired spectral radius.
The neural activations $x(n)$ have been initialized with 0.
Two attractor patterns of length 4000 have been generated and stored in p_1 and p_2. These patterns have been produced under similar starting conditions, to reduce the risk of existing bifurcations, as well as have a similar outline.

In the first phase, the network is sequentially entrained with the two patterns, using the update equation (4.1). The first 500 steps of both runs have been discarded, to account for initial state washout.

In the next phase, the following matrices and vectors are defined:

- X_i, the non-discarded network activations for pattern p_i: $X_i(:,n) = x_i(n+500)$
- $X = [X_1|X_2]$
- $\widetilde{X_i}$, the non-discarded network activations, one step ahead of X_i: $X_i(:,n) = x_i(n+499)$.
- $\widetilde{X} = [\widetilde{X_1}|\widetilde{X_2}]$
- P, which concatenates all patterns (in this case two): $P = [p_1|p_2]$

The weights W^{out} are computed using ridge regression with parameter $\varrho^{out} = 0.01$:

$$W^{out} = ((XX^T + \varrho^{out} I_{N\times N})^{-1} XP^T)^T \tag{4.3}$$

In order to compute the weights of W, the following ridge regression solution results:

$$W = ((\tilde{X}\tilde{X}^T + \varrho^W I_{N\times N})^{-1} \tilde{X}(tanh^{-1}(X) - B)), \tag{4.4}$$

where ϱ^W is set to 0.01, and B is a $500 \times (2 * 3500)$ matrix which duplicates vector b.

The next step consists in computing the conceptor matrices C_1 and C_2 (eq. 3.1), using aperture $\alpha = 1000$, as well as the output signals resulting from exciting the network with the respective pattens :

$$x(n+1) = C_i \tanh(\, Wx(n) + b) \tag{4.5}$$

Varying μ from -0.5 to 1.5 in steps of 0.01, morphed signals of length $500 * 512$ are generated:

$$x(n+1) = (\mu C_1 + (1-\mu)C_2) \tanh(\, Wx(n) + b) \tag{4.6}$$

Fast Fourier Transform (FFT) is performed on each 512 - sized chunk (see section 4.2 for details on FFT), and then the average over all chunks is computed and compared to the averaged signals for the parent patterns, achieved in the same manner.

4.1.1 Normalized Root Mean Square Error

In order to evaluate how well a signal of length L has been learned by the network, the output signal $y(n)$ will be matched against the original signal, $p(n)$, using normalized root mean square error:

$$\sqrt{\frac{\sum_{n=1}^{L}\left(y_i(n)-p_i(n)\right)^2}{L\ \mathrm{var}(y_i-p_i)}} \tag{4.7}$$

Denote p_m a morphed pattern between p_1 and p_2. In a similar manner, NRMSE is calculated between (p_m , p_1) and (p_m , p_2), and expected that the error would be smaller for the closer pattern. The exact details of the procedure are described in 4.2.

4.1.2 Rössler Attractor

The defining ODEs for the Rössler attractor are as follows:

$$\begin{cases} dx/dt = -y - z \\ dy/dt = x + ay \\ dz/dt = b + z(x - c) \end{cases} \tag{4.8}$$

The values a, b and c are the parameters used to define the family of dynamical systems.

4.1.3 Lorenz Attractor

The Lorenz attractor is defined as satisfying the ODEs:

$$\begin{cases} dx/dt = \sigma(y - x) \\ dy/dt = rx - y - xz \\ dz/dt = xy - bz \end{cases} \tag{4.9}$$

Similarly to the other attractor, σ, r and b are the defining parameters of the system.

In order to generate points in the attractors, one starts with an initial point $p(0)$, and applies an iterative solution to the system:

$$p(t + \Delta t) = p(t) + dx/dt\Delta t + dy/dt\Delta t + dz/dt\Delta t \tag{4.10}$$

Since we are working with integer time steps, the equation is reduced to:

$$p(t + 1) = p(t) + dx/dt + dy/dt + dz/dt \tag{4.11}$$

For every step, the derivatives of each coordinate with respect to time are added to their respective coordinate value. This has been followed in the code implementation of the attractor pattern generation, with the addition that much more points are being generated, and only a fraction of them are retained in the final pattern. This step is necessary because the attractor is chaotic, hence sensitive to small changes on the long run, and we desire patterns that are not affected by the choice of time difference Δt, since the conversion from the con-

tinuous system to the discrete does not preserve the attractor perfectly. For this reason, this value is kept constant, and only the retention percentage of the generated points is changed.

4.2 Quantifying morphing

In order to assess the results of pattern morphing, Fourier analysis technique is used. It is important to note that the attractors used for the experiments are not periodic, but exhibit low energy and high energy states, with arbitrary jumps from one state to the other. Therefore, comparing two related signals proves to be challenging, simply because the samples may not match. Figure (4.1) illustrates how different two samples of the same signal can be.

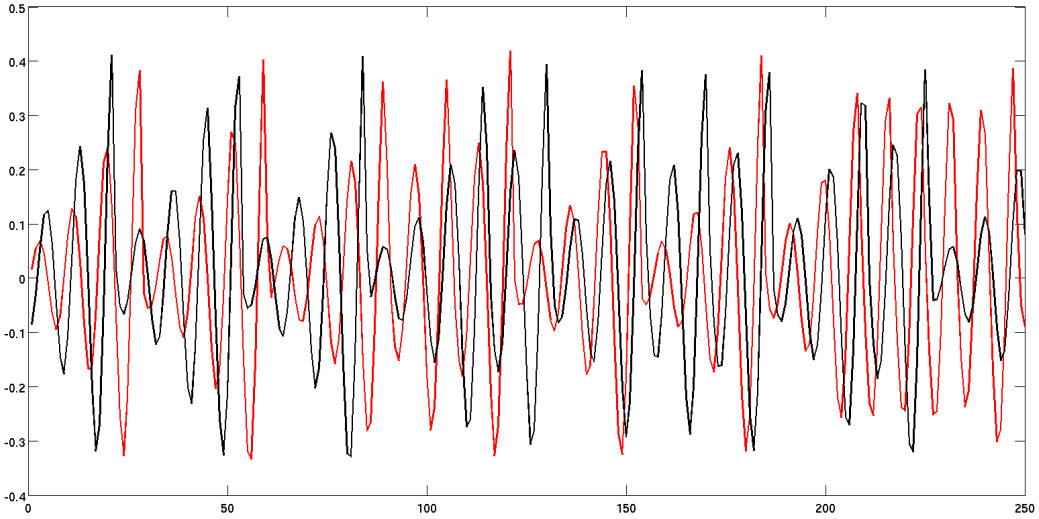


Fig. 4.1: One dimension of a section of a Rössler attractor with the length of 500 timesteps. The red signal represents the first half of the signal, while the the black represents the second. Seemingly arbitrary jumps from low-amplitude oscillations to high-amplitude oscillations and back can be spotted.

The two signals in the figure have been generated using the same attractor parameters, with different starting points. Ideally, they would be identified as coming from the same source, hence returning a small comparison error. However, comparing the data points directly proves to be a disaster. This is where Fourier analysis comes into play. I will summarize the some elementary notions found in any textbook introducing Fourier transforms.

Fourier transform is an invertible, linear transformation which converts a signal f to its frequency components, which are of sinusoidal nature.

$$f(v) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi i v t} dt \tag{4.12}$$

The discrete Fourier transform is the generalized transformation to discrete time, and converts a list of equally spaced N samples f_k of a signal into the list of coefficients of a finite combination of complex sinusoids F_k:

$$F_k = \sum_{i=0}^{N-1} f_n \cdot e^{-2\pi ikn/N} \tag{4.13}$$

We can define its inverse transform as well:

$$f_k = \frac{1}{N} \sum_{i=0}^{N-1} F_n \cdot e^{2\pi ikn/N} \tag{4.14}$$

Having such a pair of transforms at our disposal allows us to compare the attractor signals in the frequency domain. This way the periodicities of the input signals are highlighted, which is exactly what we are trying to exploit when comparing two signals of similar nature. Working in the frequency domain of a signal proves to be indeed a better approach in our attempt to compare signals. However, there are still discrepancies and fluctuations when comparing signals in the raw form. In order to combat this, 2 methods have been considered: smoothing and averaging.

The main idea in smoothing is to apply a smoothing filter in the frequency domain (for instance convolve with a Gaussian or binomial kernel) and then compare the resulting frequency domain signal. However, this method proved disruptive on our data and has been discarded.

Averaging takes a sufficient number of chunks of a given size coming from the same attractor, converts them to the frequency domain, taking their absolute value, and then uses their averaged sum for comparison. Multiple experiments were run in order to determine the optimal number of chunks to be used. That number was settled for 500, bringing a balance between reliability of the error value and time efficiency.

One can see in fig. 4.2 an example of two versions of the same signal in the frequency domain at two points in time (right), as well as an averaged over 500 steps version in the frequency domain. Larger images can be found at: 6.1, 6.2, and 6.3.

The two discrete Fourier transforms can be efficiently computed using the algorithm Fast Fourier Transform (FFT) in $O(n \log n)$.

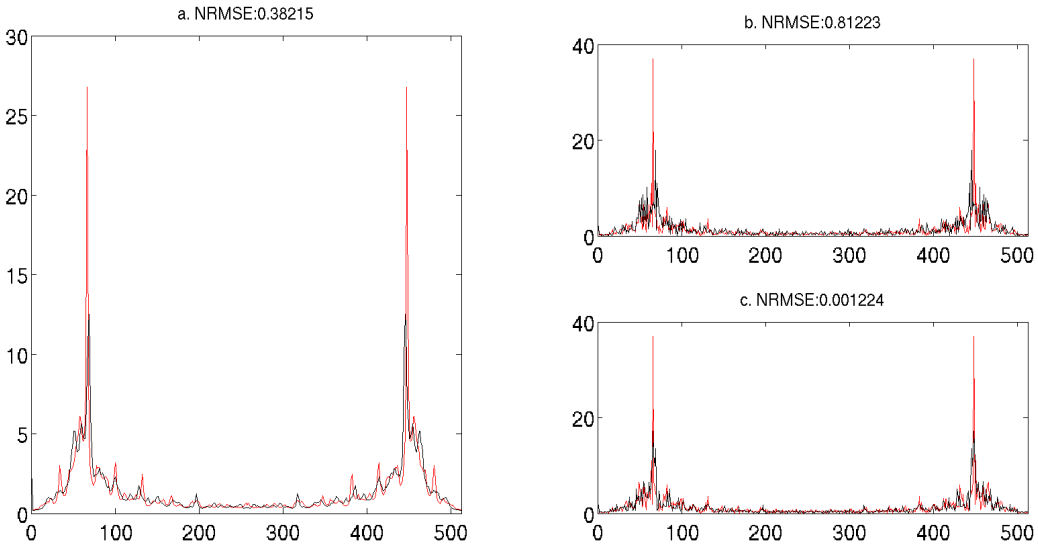


Fig. 4.2: Frequency domain visualizations for two 512-length 1D Röesler signals with similar conditions. a) represents the frequency domain averages over 500 portions of the signals, with resulting NRMSE of 0.38215. b) and c) represent two random non-averaged selections of the signals, one with NRMSE above the average (0.81223) and the other one with NRMSE below (0.001224).

4.3 Speed Morphing of Human Gaits

Speed morphing of human gaits using motion capture data poses its own challenges. Since the technical difficulties involved in processing the data have already been overcome by H. Jaeger [Jae14a], whose code I adapted to suit the purpose of this experiment, I will merely summarize the important details of the process.

First of all, data which represents walking gaits needs to be obtained, understood and processed. The data collected using motion capture data represents joint movements of selected parts of the body, during the execution a particular motion: in our case, walking. The number of joints recorded was reduced to better suit our needs. Additional effort had to be put into ensuring that data obtained from different participants was compatible.

Once the data has been processed, the network has to be trained, and a conceptor for each motion generated. Proceeding with morphing is possible from this step.

The final step is converting the network output into meaningful data. This was a two-step process: in the first part, plots portraying the position of a stick-man at a given point in time are generated. Then all these images are merged in a single video played in a chronological order, ready for inspection.

Experimental Results

5

In this section I present the results of the experiments, as well as further technical details, where needed.

5.1 Morphing two Rössler Patterns with Different System Parameters

For the first experiment, two Röessler attractor patterns with similar ODE parameters have been used. Let us call the two patterns p_1 and p_2. The values used for generating the patterns, along with important parameters used in the implementation are presented in the Appendix (6).

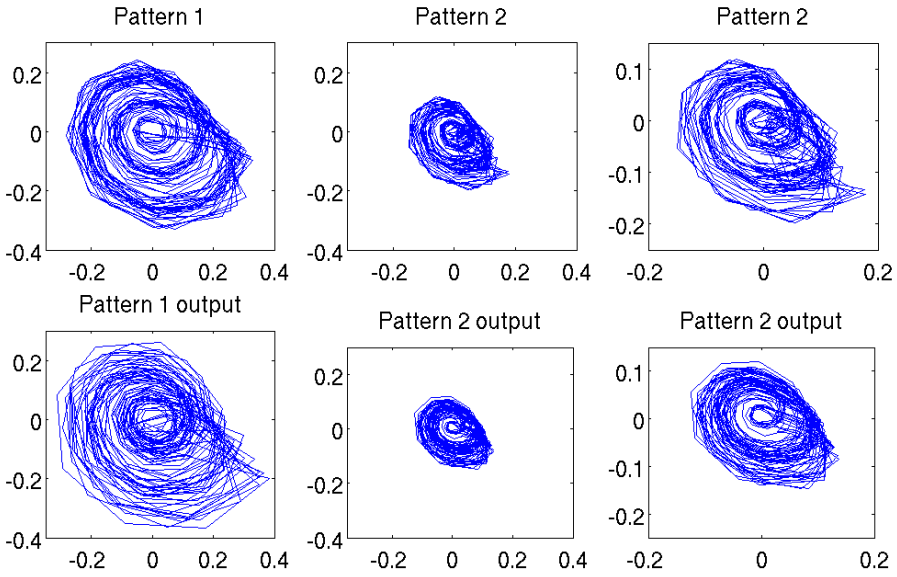


Fig. 5.1: The two patterns learned by the network. On first line we can see the initial patterns used for training: p_1 and p_2. The second line displays the output of the network y_1 and y_2, computed after re-loading the patterns using the conceptor update equation 3.7.

Following an aperture adjustment to 1000, the NRMSE errors present in the training process are shown in the table below (5.1). These errors reflect how well the network's representation of the pattern coincides with the original pattern for 3500 time steps.

The score for the first pattern denotes high degree of accuracy, which doesn't seem to be the case for the second pattern. However, if one inspects figure 5.1, the network representation of the second pattern is less noisy compared to its original, and sufficient for our experiment:

Tab. 5.1: Training errors in the first experiment

	Training Error
Pattern 1	0.0724
Pattern 2	6.8876

determining how well the two patterns can be morphed. This is the case because the comparisons are made using the network's output $y_i(n)$ rather than the initial signal $p_i(n)$.

Upon completion of the training phase, two conceptor matrices C_1 and C_2, corresponding to the initial patterns p_1 and p_2, resulted, along with a weight matrix W. In order to asses the evolution of the morphing error relative to the two patterns, 201 morphed signals have been generated, all computed using the conceptor morphing formula: $C_m = \mu C_1 + (1-\mu)C_2$, $y_m = W^{out}x$, for $\mu \in [-0.5, 1.5]$, taken in increments of 0.01 .

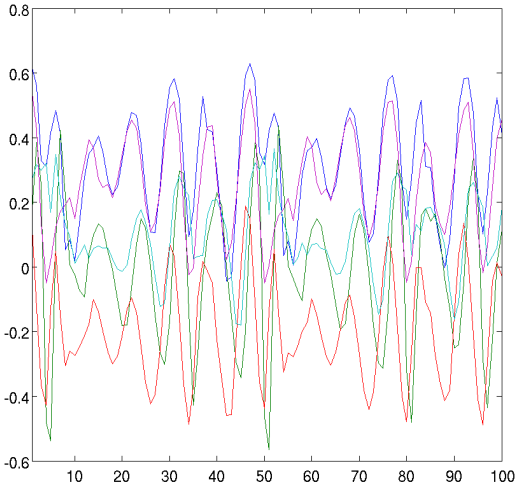


Fig. 5.2: Selected 100-step activations of the network during morphing. The neurons oscillate in a specific range, without being overexcited or "stepping on each others' toes".

Examining figure 5.3, one can see that as the morphed signal moves towards a specific pattern, its error with respect to that pattern decreases, while the error with respect to the other pattern increases. This was intuitively expected, and desired to say the least. The two lines cross at 0.4/0.6, which it fairly close to the middle.

Typical examples of network behaviour can be seen in fig. 5.2, 6.4 and 6.5 (the last two in the Appendix). Fig. 5.2 illustrates typical neural activations in a successfully-trained network. Fig 6.4 portrays the two patterns alongside one of the morphed patterns, while fig. 6.5 reveals an averaged frequency domain representation of the same morphed signal, compared to the frequency domain versions of the original patterns.

Proceeding with the inspection of the graphs' irregularities, there exist two noteworthy peaks (fig. 5.4), which, aside from momentarily dropping the errors by a considerable amount, they accompany a likewise temporary line twist. This causes pattern p_2 to appear closer to the morphed pattern than p_1. Furthermore, the central peak occurs roughly where the two lines

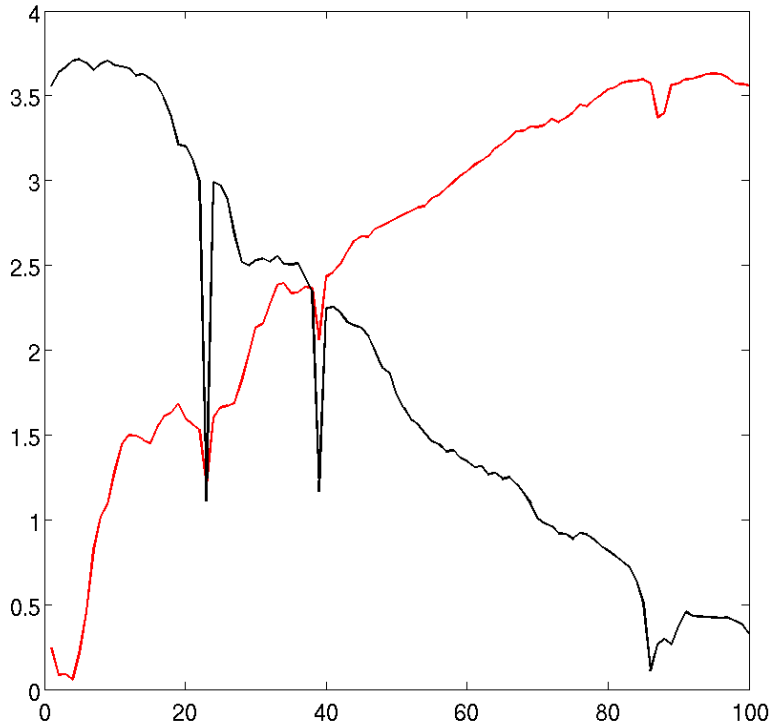
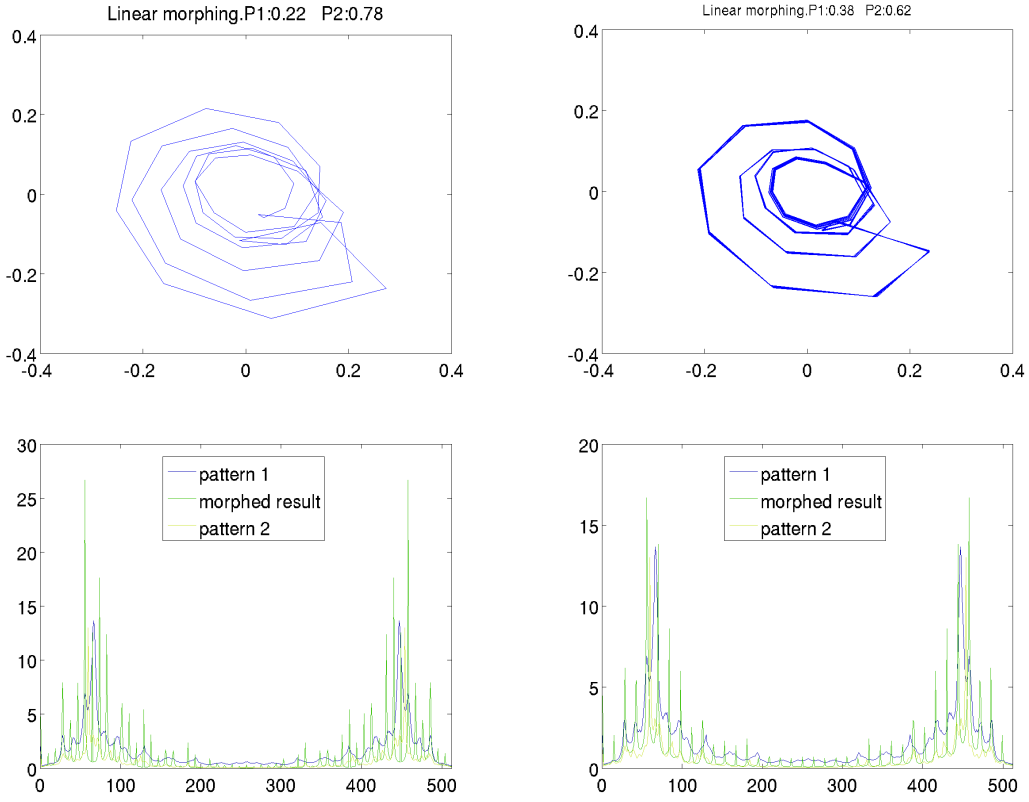


Fig. 5.3: NRMSE plots of the two Rössler patterns and their morphed signals. The red line indicates NRMSE between y_m and y_1, while the black shows NRMSE between y_m and y_2. μ varies between 0 and 1 in step sized of 0.01, which means that only interpolation is present.

would meet if there were no perturbations. In both cases, the following points in the graph follow the increase/decrease expected before the perturbation occurred. This indicates robustness to small perturbations, yet at the same time propensity for such events, since every μ can be subject to attracting/repelling behaviour.

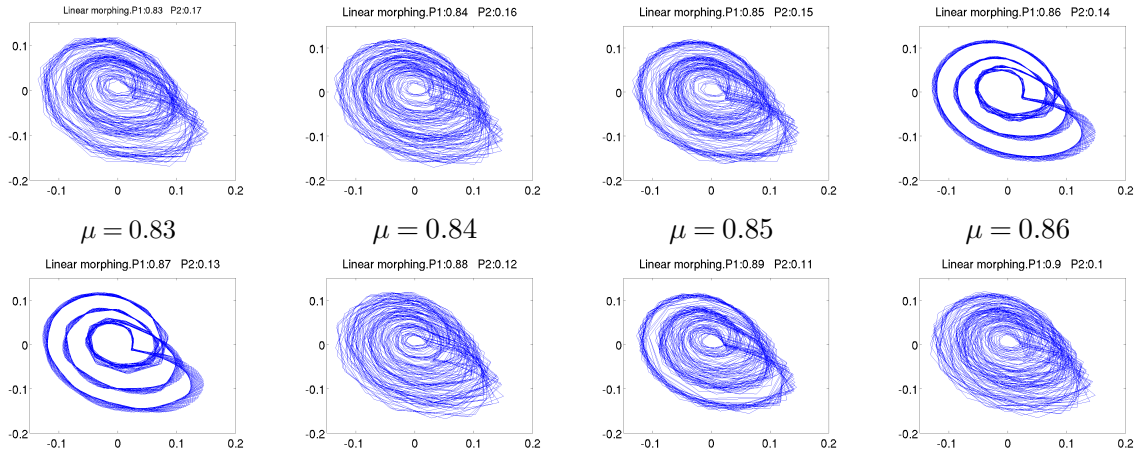
Overall, the second pattern appears to have a linear decrease, corresponding to a linear increase in μ. The first pattern, on the other hand, though mostly increasing, exhibits a more accelerate increase in the beginning, only to slow down as the morphed pattern converges to it.

Another noteworthy phenomenon is present between abscissae 86 and 91, where both functions exhibit a sudden drop in values, better expressed by the second pattern (fig. **??**). Unlike the previously discussed 1-timsestep peaks, these drops are longer-lived and are found considerably closer to either pattern (in this case, p_1).



$\mu = 0.22$ $\mu = 0.38$

Fig. 5.4: The two peaks where, despite $\mu < 0.5$, the second pattern is closer to the morphed pattern. On the top tow, the last 500 steps of the signals produced are displayed (steps from 256013 to 256512). On the bottom row, the frequency domain representation of the averages is presented. We observe that the two patterns barely retain the shape of a Rössler attractor, therefore the network failed to produce a true morphed pattern for those values.



$\mu = 0.83$ $\mu = 0.84$ $\mu = 0.85$ $\mu = 0.86$

$\mu = 0.87$ $\mu = 0.87$ $\mu = 0.88$ $\mu = 0.9$

Fig. 5.5: Morphed patterns for μ ranging between 0.83 and 0.90.

5.2 Speed morphing two Lorenz attractors

In the second experiment, two Lorenz patterns with identical attractor conditions and different paces were morphed. More precisely, the patterns were generated under the same conditions, only that the first pattern (p_1 saved one point for every 20 points generated, while the second pattern (p_2 saved one point out of every 15). This means, in turn, that the actual time difference between two points is 0.1 units for p_1 and 0.075 units for p_2. Section 6 (found in the Appendix) presents the initial conditions used to generate the patterns, along with important parameters in training. The training errors were: 1.034053 in the case of the first pattern, and 1.708271 for the second.

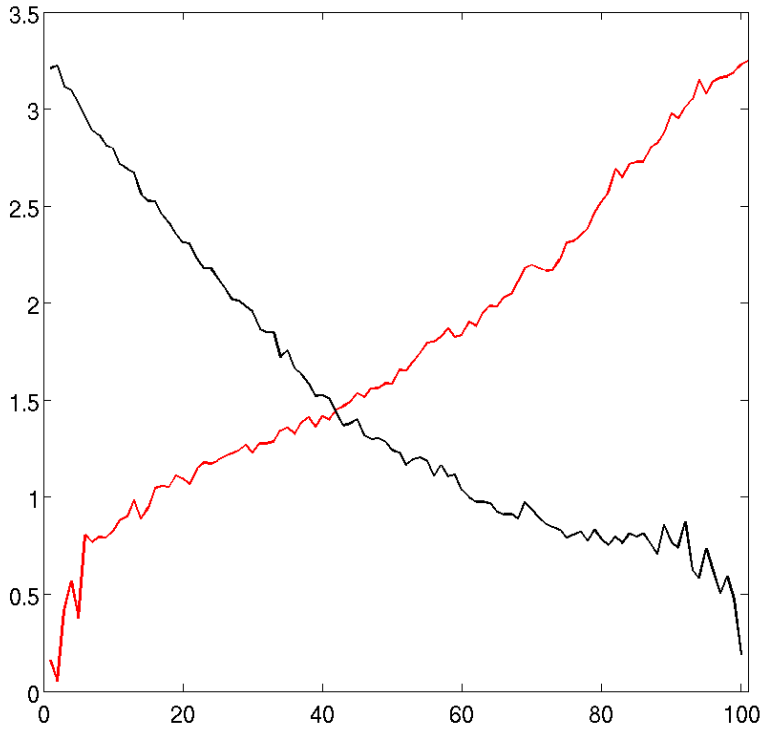


Fig. 5.6: NRMSE plots of the two Lorenz patterns and their morphed signals. The red line indicates NRMSE between y_m and y_1, while the black shows NRMSE between y_m and y_2. μ varies between 0 and 1 in step sizes of 0.01.

Similar to the first experiment, figure 5.6 represents the NRMSE values in the frequency domain between the morphed patterns and the reference patterns p_1 and p_2.

Analysing the plot, the errors obtained in this experiment seem to follow a much better-behaved overall trajectory than the first experiment, lacking striking perturbations. However, the shape is rather jagged, which indicates that the path between the two patterns described by morphing zig-zags extensively, but it could simply be due to the sampling/measurement technique.

One interesting aspect of the shape of the plot is found when the morphed patterns approach the original patterns, for μ values close to 0 or 1. The error climb follows a slightly different trajectory locally to those points, increasing abruptly as the morphed pattern diverges from the nearby pattern. Additionally, the shape is more rigged in the affected area. It is even more remarkable that nothing special appears to happen from the point of view of the non-local pattern.

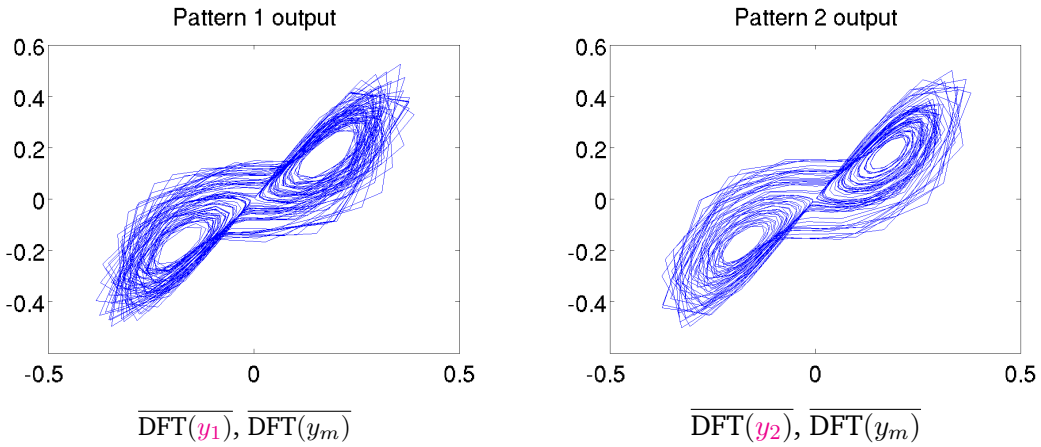


Fig. 5.7: A sequence of 500 steps of the learned patterns, as outputted by the network.

When extrapolation is introduces in the equation, the results suddenly become unreliable,as it is the case here. Figure 5.8 stretches the current morphing span by 10 steps to the left and 10 others to the right, causing μ to vary from -0.1 to 1.1.

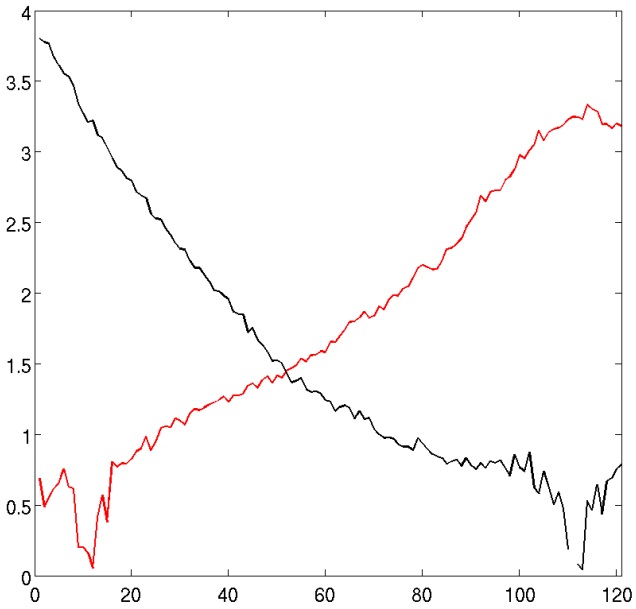


Fig. 5.8: NRMSE plots of the two Lorenz patterns and their morphed signals. The red line indicates NRMSE between y_m and y_1, while the black shows NRMSE between y_m and y_2. μ varies between 0 and 1 in step sizes of 0.01.

5.3 Morphing Human Gaits

Following the set-up and code of Prof. Herbert Jaeger, I attempted at a linear morphing between two walking patterns by the same person, one of which had a slower pace. The training data was taken from the CMU Mocap repository and inserted into a network for training. The purpose of the experiment was to inspect whether speed morphing is possible in a raw form of the data.

The resulting video characterizing the output of the network can be found here [https://www.youtube.com/watch?v=l5thk9V74H0]. On the upper left corned of the screen, three bars can be seen, out of which only two are of interest for us: the blue and the green ones. The blue corresponds to the normal walking gait, while the green represents the slower walking gait. Whenever a bar is full, it means that the respective pattern is being called. Whenever more than one bar is present, the motion displayed is a morph between the two patterns, with coefficients corresponding to the heights of the bars. The motion starts with a normal walking motion, then it slowly transitions to a slower motion, only to revert back to the normal gait.

As it can be seen from the video, the results were only partly satisfactory. If run for a short amount of time, the transition from one pattern to the other appears natural. However, if one tries to inspect the transitions at a slower pace, it becomes clear that the network fails to produce a true intermediate pattern. The character is slowly rotating during this transition, which, in my opinion, indicates that the two patterns are out of phase in regards to the moving leg. The slow walking gait spends more time on each leg, which causes the moving leg to differ after a few steps. At this point, the network is confused, and takes many tiny steps in order to resynchronize in turns with either pattern. More intriguingly, the network appears to slide on invisible ice in some portions. Under these circumstances, the very idea of "natural intermediate walking" seems an unsafe path to pursue.

Throughout the video, the mistakes the network made in its attempt to morph patterns are connected to the fact that the network does not appear to comprehend the concept of speed, but merely rely on physical clues regarding member's movement. An insurmountable amount of reasons could sit behind this disappointing behaviour, including the possibility that the network is simply not capable speed morphing in such a raw form. Looking at the bright side, the network did produce intermediate walking for some periods of time, and might be able to produce better results if it learned the concept of speed. Further research will have to be conducted in this area to give a answer of if and how speed morphing could be achieved with this architecture.

Conclusion

6

Pattern morphing on attractors proved to be possible, yet not flawless. In the first experiment, there were various intermediate points with abrupt changes in behaviour, only to return to its previous trajectory after a few iterations. It is always hard to determine whether these perturbations are the product of floating point errors, or whether attractor dynamics came into play. These doubts are amplified by the lack of powerful tools to diagnose the behaviour, both conceptually in order to establish what good morphing really is, as well as numerically, since the named patterns are highly non-linear and non-periodic.

Further research could involve crafting better tools more suitable to the task: sensitive to fine details in the pattern, insensitive to noise, understanding of the attractors' "jumps" in behaviour. Finding the dynamical path followed by conceptor morphing could also be a enlightening endeavour, attempted as it may be mathematically or computationally. It would be able explain the curious shape of the morphing curve, and the mathematical meaning of a morphed pattern in our context.

A verdict of speed morphing on motion capture data proved to be a challenge. On the one hand, the experiment was not a complete fiasco, which gives us some hope in a potential comeback. On the other hand, the network failed to comprehend the very nature of its given task, which was disappointing, to say the least. Nevertheless, during the second experiment, in which speed morphing was attempted on Lorenz attractor, the results with satisfactory enough to give hope that speed morphing on motion capture data might be realizable.

Appendix

Frequency domain averaging

Figures 6.1, 6.2, and 6.3 display two portions of the same original Rössler signal (red and black) in the frequency domain. The length of the signals is 512 in all cases, and the NRMSE error calculated in the frequency domain is presented above. The first figure (6.1) is an average in the frequency domain over 500 samples, while the last two figures display rather extreme values of NRMSE, which illustrates just how necessary averaging is.

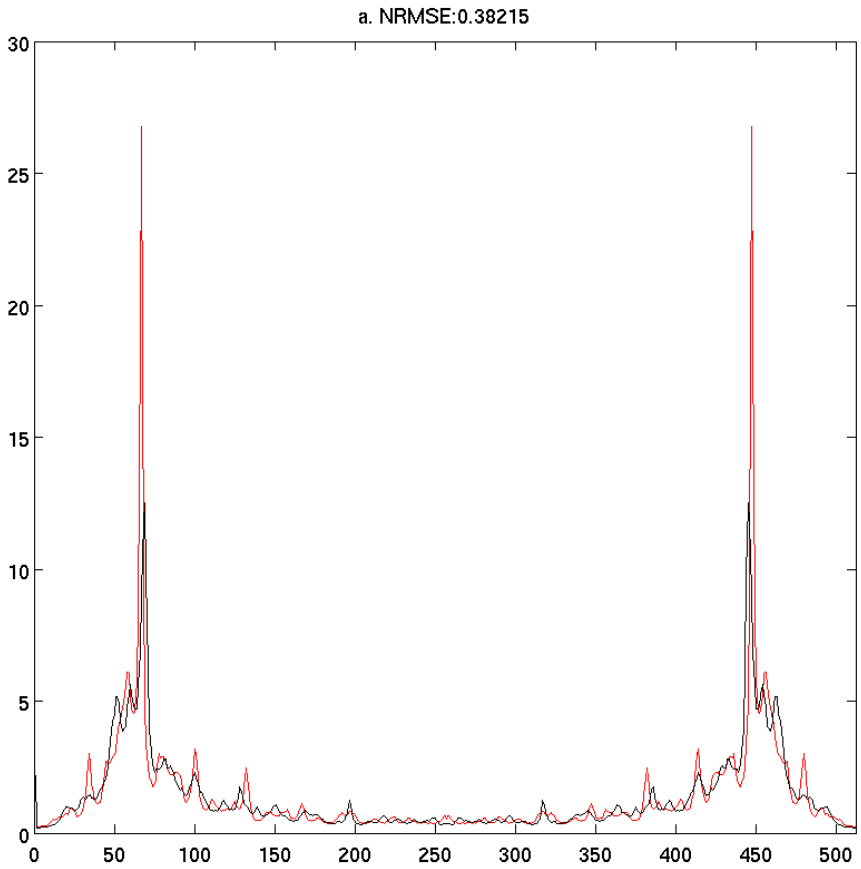


Fig. 6.1: Example of frequency domain averages of 500. Resulting NRMSE of 0.38215.

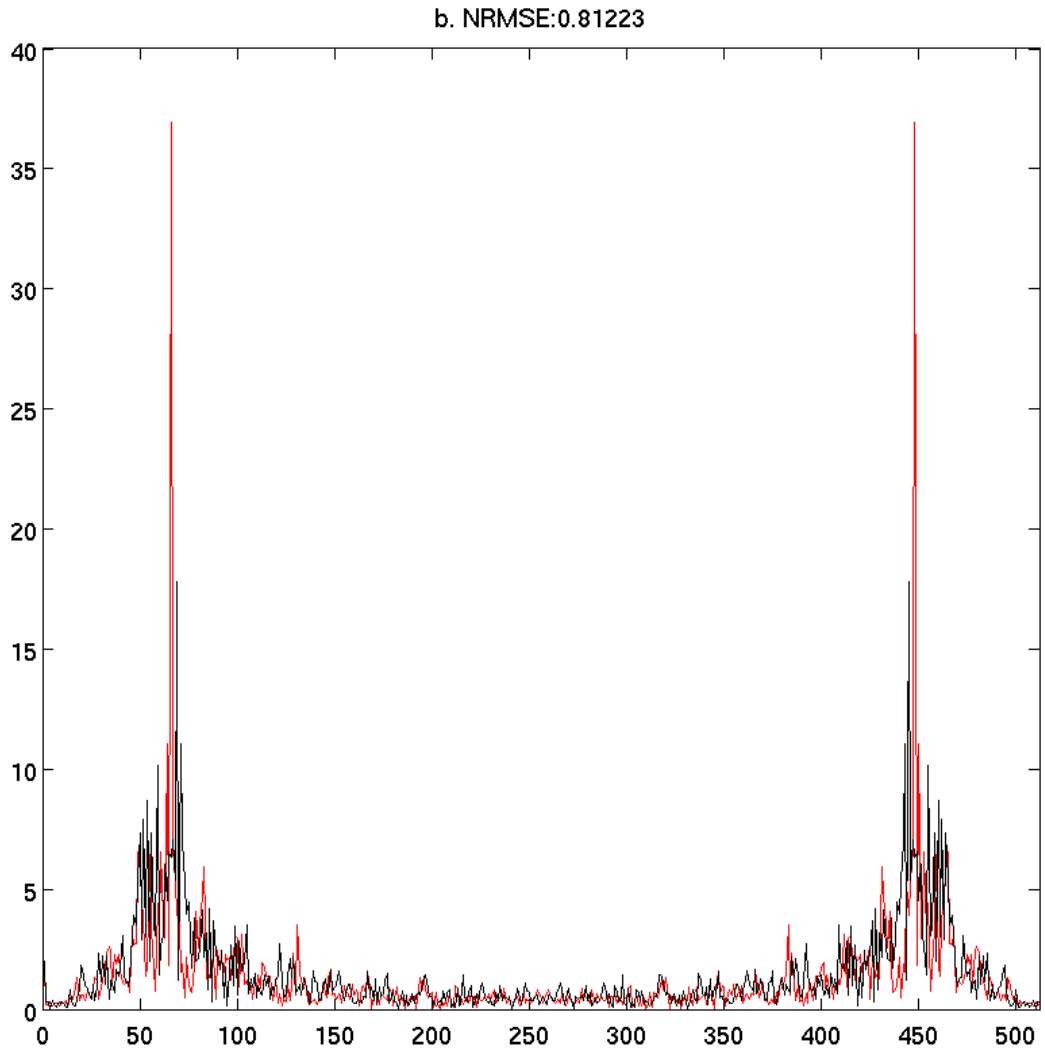


Fig. 6.2: Example of frequency domain representation of a signal. Resulting NRMSE of 0.81223, which is above the average of 0.38215 obtained when averaging over 500 such portions.

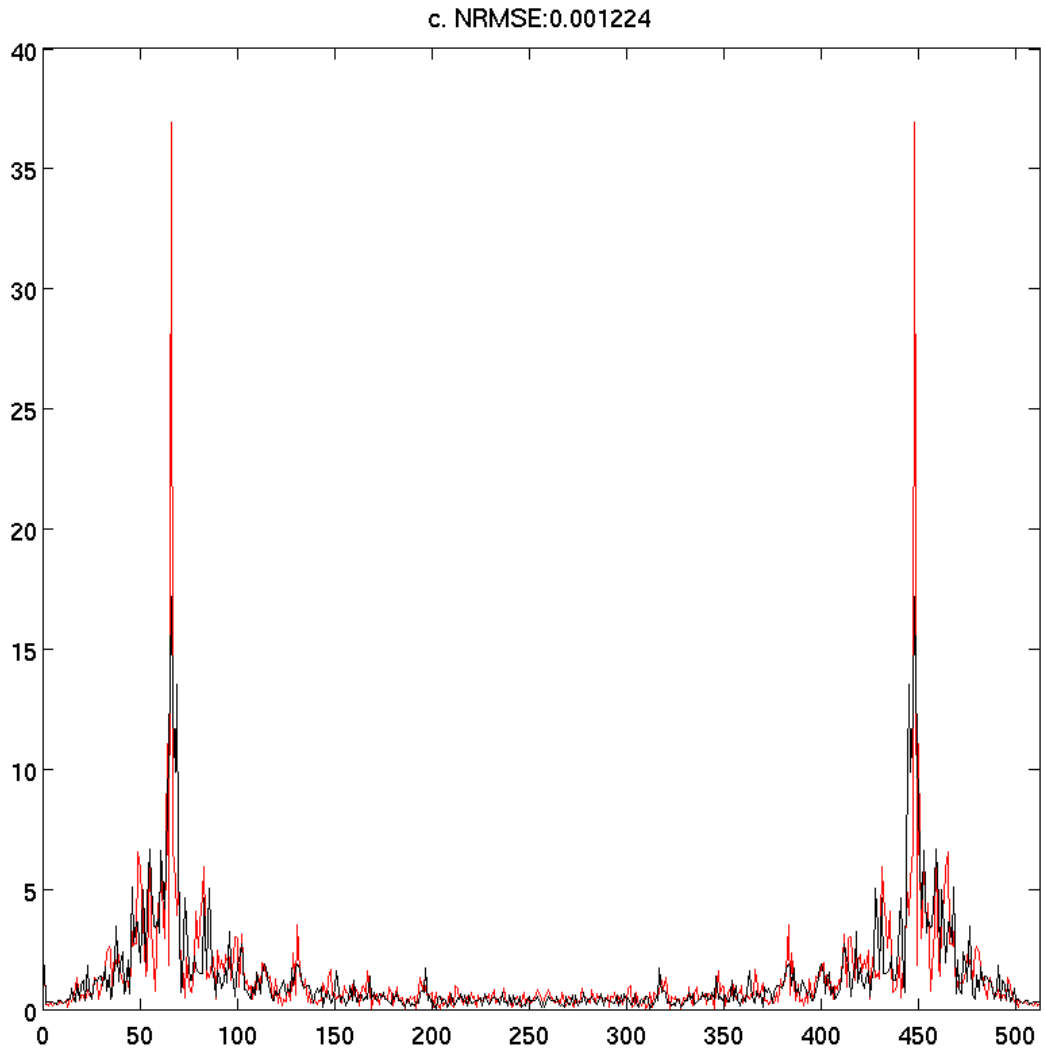


Fig. 6.3: Example of frequency domain representation of a signal. Resulting NRMSE of 0.001224, which is below the average of 0.38215 obtained when averaging over 500 such portions.

Experiment 1: Initial Conditions

Pattern	Starting point coordinates			Attractor parameters			Δ
	x	y	z	σ	ρ	β	
p_1	0.5943	−2.2038	0.0260	0.2	0.2	8	0.05
p_2	0.5943	−2.2038	0.0260	0.3	0.2	8	0.05

Tab. 6.1: Pattern generation parameters for Rössler experiment.

Aperture	1000
Bias scaling	0.2
Spectral radius	1
W_{in} scaling	0.7
Generated pattern length	4000
Washout length	500
Number of input data dimensions	2
Number of internal units	500
ρ_W	0.01
ρ_{out}	0.01

Tab. 6.2: Algorithm parameters for Rössler experiment.

With regards to the above table, ρ_W and ρ_{out} are Tikhonov regularization parameters, used for calculating W using formula 4.4; respectively W_{out} using formula 4.3.

The following figure displays the original patterns in the first experiment (margin) alongside the morphed pattern (center). Like the previous figures, $\mu = 0.14$.

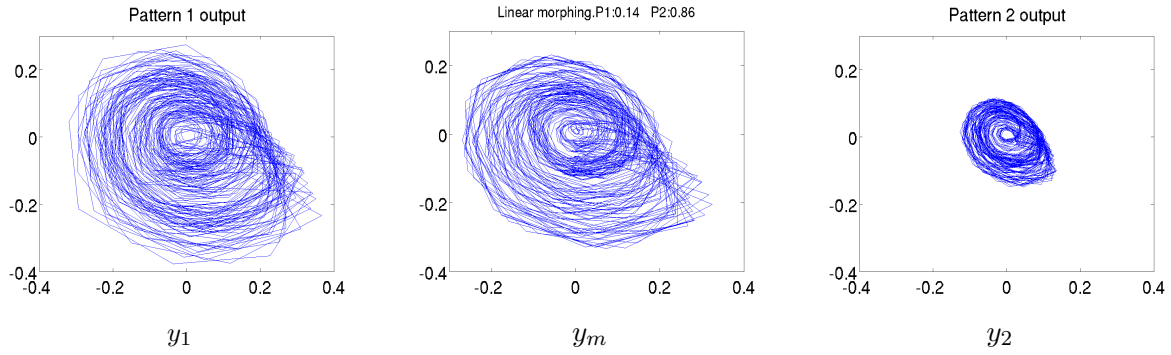


Fig. 6.4: Last 500 steps of y_m for $\mu = 0.14$, next to y_1 (left) and y_2 (right).

Analysing morphed Rössler signals

The following two figures (6.5, 6.6) portray frequency domain representations of the two Rössler patterns in Experiment 1, as well as a morphed signal obtained for $\mu = 0.14$. The outline of the morphed pattern follows the first pattern closely, much more so than the second pattern.

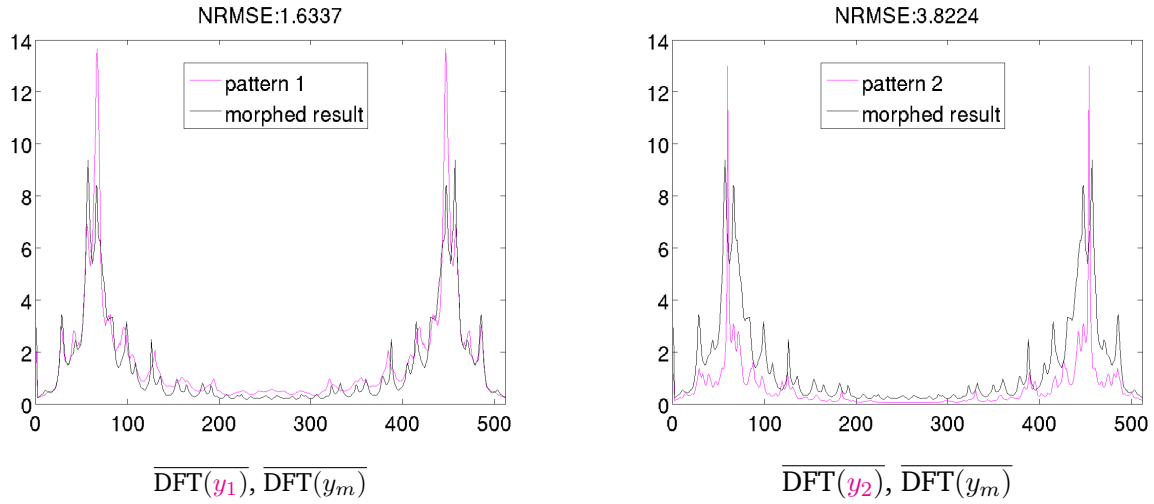


Fig. 6.5: Plots in the frequency domain of the averaged discrete Fourier transforms (DFT).

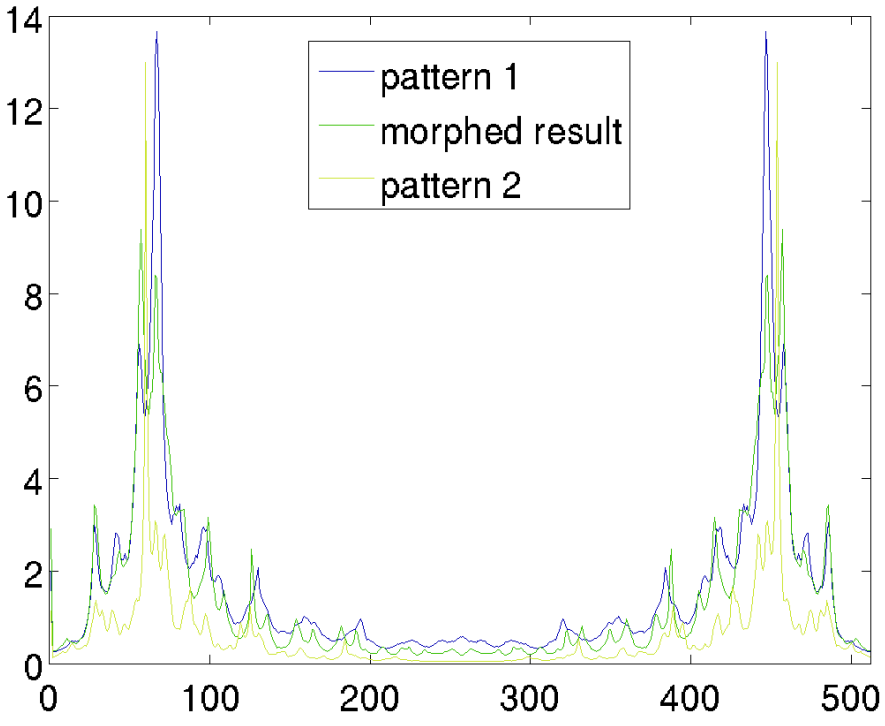


Fig. 6.6: Plots in the frequency domain of the averaged discrete Fourier transforms (DFT).

Experiment 2: Initial Conditions

Starting point coordinates			Attractor parameters			Δ
x	y	z	σ	ρ	β	
10.036677794959058	9.98674414052542	29.024692318601613	10	28	$8/3$	0.05

Tab. 6.3: Pattern generation parameters for Lorenz experiment.

Aperture	100
Bias scaling	0.2
Spectral radius	1
W_{in} scaling	2.5
Generated pattern length	4000
Washout length	500
Number of input data dimensions	2
Number of internal units	500
ρ_W	0.0001
ρ_{out}	0.01

Tab. 6.4: Algorithm parameters for Lorenz experiment.

Analysing morphed Lorenz signals

Figures (6.7, 6.8) portray frequency domain representations of the two Lorenz patterns in Experiment 2, as well as a morphed signal obtained for $\mu = 0.09$. They are plotted for only one of the two dimensions, as FFT was applied individually.

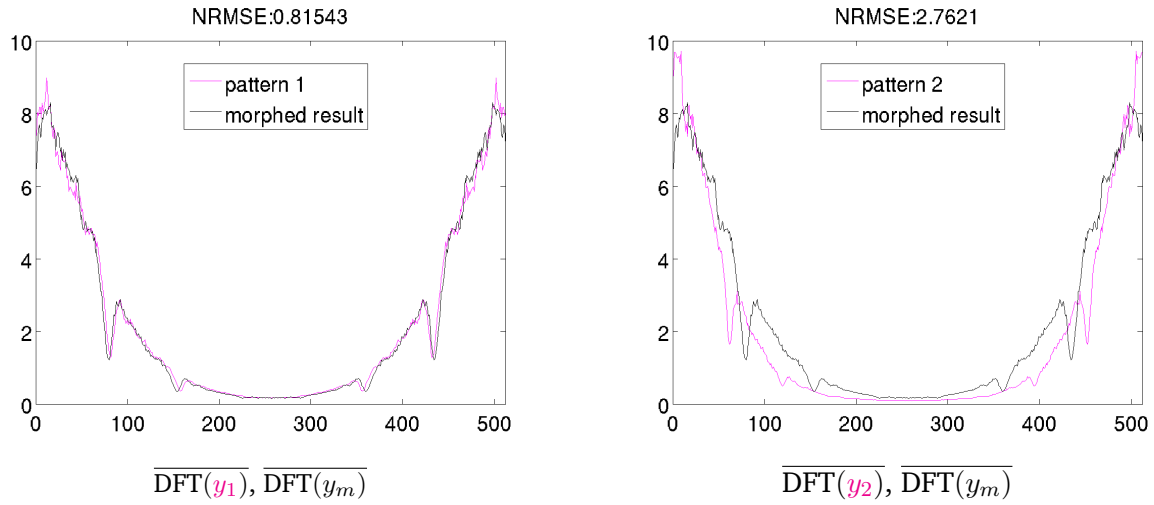


Fig. 6.7: Plots in the frequency domain of the averaged discrete Fourier transforms (DFT).

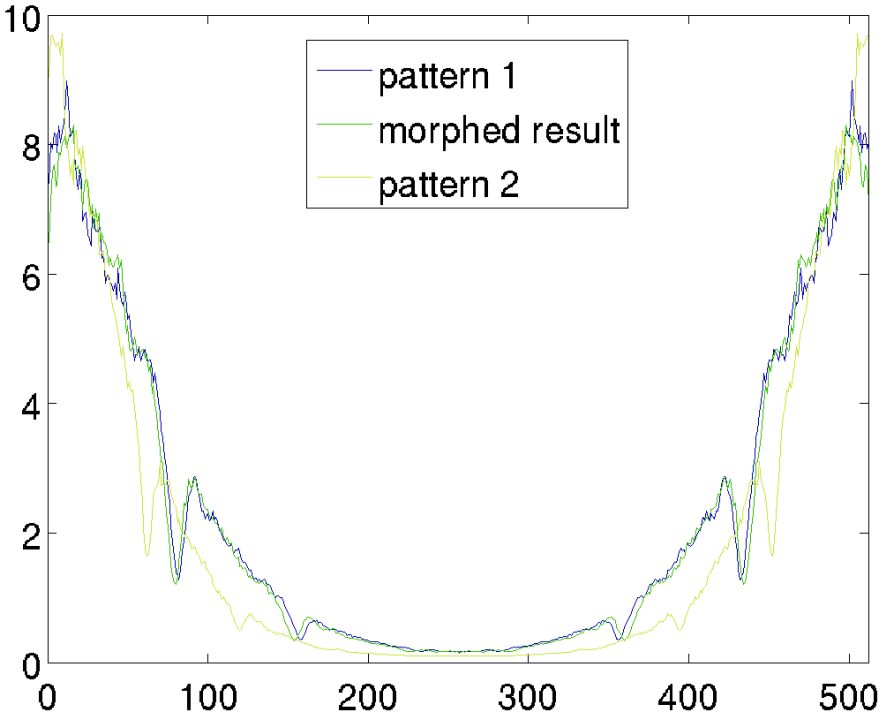


Fig. 6.8: Plots in the frequency domain of the averaged discrete Fourier transforms (DFT) in one dimension of the signal.

Bibliography

[Aja+13] M. Ajallooeian, J. van den Kieboom, A. Mukovskiy, M. A. Giese, and A. J. Ijspeert. „A General Family of Morphed Nonlinear Phase Oscillators with Arbitrary Limit Cycle Shape“. In: *Physica D: Nonlinear Phenomena* 263 (2013), pp. 41–56 (cit. on p. 4).

[Alb+92] D. Albesano, R. Gemello, and F. Mana. „Word recognition with recurrent network automata“. In: *Neural Networks, 1992. IJCNN., International Joint Conference on*. Vol. 2. 1992, 308–313 vol.2 (cit. on p. 2).

[Ben+94] Y. Bengio, P. Simard, and P. Frasconi. „Learning long-term dependencies with gradient descent is difficult“. In: *Neural Networks, IEEE Transactions on* 5.2 (1994), pp. 157–166 (cit. on p. 4).

[Bos+13] K. J. Boström, H. Wagner, M. Prieske, and M. de Lussanet. „Model for a flexible motor memory based on a self-active recurrent neural network“. In: *Human Movement Science* 32 (2013), pp. 880–898 (cit. on p. 4).

[Doy93] K. Doya. „Bifurcations of Recurrent Neural Networks in Gradient Descent Learning“. In: *IEEE Transactions on Neural Networks* 1 (1993), pp. 75–80 (cit. on p. 4).

[FN93] K. Funahashi and Y. Nakamura. „Approximation of dynamical systems by continuous time recurrent neural networks“. In: *Neural networks* 6.6 (1993), pp. 801–806 (cit. on pp. 2, 6).

[IT04] M. Ito and J. Tani. „Generalization in learning multiple temporal patterns using rnnpb“. In: *Neural Information Processing*. Springer. 2004, pp. 592–598 (cit. on p. 2).

[Jae+12] H. Jaeger, M. Ajallooeian, A. Billard, et al. *Technical report on dynamic extensibility methods*. Tech. rep. Jacobs University Bremen, 2012 (cit. on p. 4).

[Jae01] H. Jaeger. „The “echo state” approach to analysing and training recurrent neural networks-with an erratum note“. In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148 (2001), p. 34 (cit. on pp. 2, 4).

[Jae02] H. Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002 (cit. on pp. 4, 6).

[Jae14a] H. Jaeger. „Conceptors: an easy introduction“. In: *Computing Research Repository* abs/1406.2671 (2014) (cit. on pp. 2, 17).

[Jae14b] H. Jaeger. „Controlling Recurrent Neural Networks by Conceptors“. In: *Computing Research Repository* abs/1403.3369 (2014) (cit. on pp. 2, 3).

[Kar+92] T.W. Karjala, D.M. Himmelblau, and R. Miikkulainen. „Data rectification using recurrent (Elman) neural networks“. In: *Neural Networks, 1992. IJCNN., International Joint Conference on*. Vol. 2. IEEE. 1992, pp. 901–906 (cit. on p. 2).

[MT91] K. Murakami and H. Taguchi. „Gesture Recognition Using Recurrent Neural Networks“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’91. New Orleans, Louisiana, USA: ACM, 1991, pp. 237–242 (cit. on p. 2).

[NP90] K.S. Narendra and K. Parthasarathy. „Identification and control of dynamical systems using neural networks“. In: *Neural Networks, IEEE Transactions on* 1.1 (1990), pp. 4–27 (cit. on p. 2).

[Pas+12] R. Pascanu, T. Mikolov, and Y. Bengio. „On the difficulty of training recurrent neural networks“. In: *arXiv preprint arXiv:1211.5063* (2012) (cit. on p. 4).

[ST93] Linda B Smith and Esther Ed Thelen. „A dynamic systems approach to development: Applications.“ In: *This book grew out of a workshop," Dynamic Systems in Development," held for the Society for Research in Child Development in Kansas City, KS, Apr 1989*. The MIT Press. 1993 (cit. on p. 10).

[Wer90] P.J. Werbos. „Backpropagation through time: what it does and how to do it“. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560 (cit. on p. 4).

[WJ12] J. Wright and I. Jordanov. „Intelligent approaches in locomotion“. In: *Proc. WCCI 2012 IEEE World Congress on Computational Intelligence*. 2012, pp. 1–8 (cit. on p. 4).

[WS09] F. Wyffels and B. Schrauwen. „Design of a central pattern generator using reservoir computing for learning human motion“. In: *Proc. Advanced Technologies for Enhanced Quality of Life, 2009. AT-EQUAL ’09*. IEEE, 2009, pp. 118–122 (cit. on pp. 2, 4).

[Wyf+14] F. Wyffels, J. Li, T. Waegeman, B. Schrauwen, and H. Jaeger. „Frequency Modulation of Large Oscillatory Neural Networks“. In: *Biological Cybernetics* 108 (2014), pp. 145–157 (cit. on p. 4).

[WZ89] R. J. Williams and D. Zipser. „A learning algorithm for continually running fully recurrent neural networks“. In: *Neural computation* 1.2 (1989), pp. 270–280 (cit. on p. 4).

List of Figures

List of Tables

Acknowledgement

I would like to thank Professor Herbert Jaeger, for his continuous support and supervision. I would also like to thank Brennan Bell for the insightful discussions in dynamics.

This document uses Clean Thesis style developed by Ricardo Langner.

Declaration

With my signature, I certify that this thesis has been written by me using only the indicates resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Bremen, August 31, 2015

Alina Dima