Learning Motor Primitives with Echo State Networks

Jakob Gütschow, Johannes Lohmann, Danil Koryakin, and Martin V. Butz

Cognitive Modeling, Department of Computer Science, University of Tübingen, Sand 14, 72076 Tübingen, Germany

Abstract. Movement control based on the combination of simple motor patterns (movement primitives) has proven to be a suitable approach for modeling human and robot behavior. Learning movement primitives involves the approximation of movement trajectories that can be described in terms of non-linear functions, different techniques have been proposed to solve this task. Here we investigate the ability of a special type of recurrent neural network, so called *echo state networks*, to perform this learning task. Our results based on a simple 2D simulation show that even quite small and simple structured networks are able to reproduce complex motion patterns.

Keywords: Movement Primitives, ESN, Reservoir computing

1 Introduction

The planning and execution of precise reaching movements is a highly complex challenge in robotics. Due to the high degrees of freedom of modern manipulators, there is a high level of redundancy inherent in these tasks. Targets can be reached on a large variety of trajectories and with a large variety of end-postures.

A promising approach to resolve this redundancy is to decompose the motions into basic patterns, so-called motor primitives [6]. The foundation for these primitives comes from biology, where they are described as “motor patterns that are considered basic units of voluntary motor control, thought to be present throughout the life-span” or simply “building blocks of movement generation” (see [11], p. 1). In vertebrates, the neuronal realization of motor primitives is likely located in the spinal cord. Here populations of neurons were found that recruit groups of muscles whose activities remained proportionally fixed throughout their recruitment [2]. By flexibly combining just a relatively small number of these building blocks, uncountable movement patterns and behaviors can be generated. It has been speculated, that such motor primitives may be crucial elements in early motor learning [2]. The existence of motor primitives implies that there might be some kind of *central pattern generator* (CPG) that combines simple low dimensional inputs to produce complex high dimensional outputs. A combination of low dimensional inputs can be achieved for instance by blending different motor primitives together.

To model such CPGs, two problems have to be addressed. First, the movement patterns have to be defined in terms of trajectories; for instance as time-series of task-space coordinates or angles in joint space. Second, a control process is necessary to monitor the movement execution and to adapt it dynamically if necessary. Apart from other approaches to learn and combine motor primitives like dynamic movement primitives [15] or Gaussian mixture models [1,10], [5] suggested an implementation via recurrent neural networks (RNNs). Especially the ability to autonomously generate rhythmic activation in a self-recurrent way is an interesting feature of RNNs as it seems to reflect the properties of biologic pattern generators.

The idea to learn motor primitives with RNNs is not new (see for instance [14]), but more recent research focuses on the capabilities of *reservoir computing*[18] to model a CPG. One architecture frequently used in studies on reservoir computing is based on the echo state networks (ESNs) presented in [7]. Compared to other RNNs, training of even very large ESNs is fast, easy to implement, and quite robust. ESNs are especially well-suited for time-series learning, and there are some examples for a successful application in the domain of movement planning and movement control. [3] compares ESNs with other frequently used algorithms used for time-series learning. [17] learned a bidirectional mapping of forward and inverse kinematics to perform goal-directed reaching movements within a single ESN. With an ESN applying leaky integrator neurons, [9] were able to reproduce trajectories like the *lazy figure eight*. Additionally, [9] also showed that the time scale of the reproduction of a certain ESN can be dynamically tuned. There have also been attempts to solve the mentioned control problem, i.e. the ability to switch between different learned motor primitives, within ESNs. As it was shown by [14], network dynamics can be shifted by bifurcation inputs. For instance, the architectures proposed by [19] and [13] proved that this approach is also viable for ESNs. Both architectures fulfill the requirements of a model for a CPG. To sum up, there is plenty of evidence for the ability of ESNs to account for the learning of movement primitives. With architectures like the one proposed by [19] or [13] it is also possible to solve the control problem within ESNs.

Even if these findings are promising, the applied networks were quite large (300 neurons in [13], and up to 2200 neurons in [19]), used special types of neurons with filter properties [20,19], or required more elaborate training mechanisms than the original ESNs [1]. Therefore, we are interested in the ability of "vanilla" ESNs to reproduce motor patterns in a completely self-recurrent way, without any additional input. As a first step, we use a simple 2D-simulator of an arm to define and execute basic movements to be learned and reproduced by the networks. We are focusing on the kinematics of the movement. Hence, we do not directly learn any dynamics. We concentrate on finding a coding scheme for motion trajectories that can be accurately learned, that can generate stable control commands, and that generalize well. Different encodings were employed

[1] For instance [19] used *ridge regression*[4], instead of the simple linear regression, proposed by [7].

to learn particular trajectories with ESNs. For instance, [13] as well as [3] used two-dimensional task space coordinates, whereas [19] learned different joint angle evolutions. In this work we assess the impact of different coding schemes on the learning performance. More precisely, we investigate if different coding schemes require different network parameters for successful learning.

As we are using networks that are guided by a constant output feedback loop without any additional external input the notion of stability is crucial. Our aim is to find *output feedback stable* networks in the sense of [16], i.e. networks that are not driven into an extreme attractor state by there own recurrent feedback. Some recent papers on output driven ESNs investigated which network properties are necessary to avoid error amplification due to recurrent feedback. While [16] pointed out that regularized[2] networks are less prone to error amplification, [12] suggested a balancing between the scaling of the output feedback strength and the amplitude of the learned time-series.

In the next section we provide a short introduction to ESNs. After this we sketch out our experimental setup. Next, we evaluate performances with respect to different coding schemes. A short discussion concludes the paper.

2 Echo State Networks

While a detailed description of the features and working of this architecture can be found in [7] and [8], this section only gives an overview of its most important features.

The basic ESN structure is quite similar to that of standard RNNs, consisting of an (optional) input layer, a layer of the hidden neurons (the so-called, dynamic reservoir), and an output layer. The crucial differences of ESNs compared to the standard RNNs are the pre-wired dynamic reservoir and the simple training procedure, which only optimizes the output weights. Figure 5 shows an overview of the architecture. Within the dynamic reservoir different dynamics unfold over time. These dynamics are combined to produce the output of the network.

It is necessary that the dynamic reservoir adheres to the so-called *echo state property*. This property is a requirement for stable dynamics and leads to a fading memory of the reservoir with respect to the input history. The *echo state property* can be achieved by restricting the spread of the reservoir weights through their division by the largest eigenvalue of the reservoir weight matrix and multiplying them with λ^*. This results in a matrix where the desired largest absolute eigenvalue equals λ^*, in other words the *spectral radius* of the matrix equals λ^*.

The state of the dynamic reservoir at time-step $n+1$ can be described by:

$$\mathbf{x}(n+1) = f(\mathbf{W^{IN}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W^{OFB}}\mathbf{y}(n)), \qquad (1)$$

where bold letters denote matrices and vectors. More precisely $\mathbf{x}(n)$ denotes the state of the dynamic reservoir at the n^{th} time-step, $\mathbf{u}(n+1)$ and $\mathbf{y}(n)$ denote the current input and the previous output, respectively. The weight matrices $\mathbf{W^{IN}}$,

[2] The term regularized refers to a low norm of the different weight matrices.

$\mathbf{W}$, and $\mathbf{W^{OFB}}$ contain weights of connections from the input neurons to the reservoir neurons, recurrent connections within the reservoir, and output feedback connections from the output neurons to the reservoir neurons, respectively. Finally, f denotes the transfer function of the units of the dynamic reservoir — usually a sigmoid. The state of the units in the output layer is obtained as follows:

$$\mathbf{y}(n+1) = f^{OUT}(\mathbf{W^{OUT}}\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n)), \tag{2}$$

where f^{OUT} denotes the transfer function of the output units — usually a linear function.

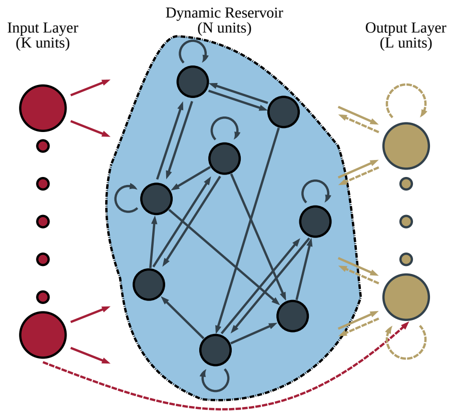

Fig. 1. Overview of the ESN-architecture. Dashed lines represent optional connections. Optimized connections are those leading to the output layer.

As noted above, a unique feature of ESNs is that the randomly initialized reservoir weights stay fixed. The only weights that are optimized during training are those connecting the dynamic reservoir and optionally the input layer with the output layer. To optimize the output weights the squared difference of the network output and the desired time-series is minimized.

The training process itself is executed in three consecutive phases. First, the initial transients of the dynamic reservoir are extinguished during a *washout phase*.

Second, the crucial part of the training is executed in the *sampling phase*. During this phase, teacher forcing is applied, feeding the intended output values back into the network via the output feedback connection weights $\mathbf{W^{OFB}}$. The resulting excitation of the reservoir is recorded in a matrix M (time-steps $\times$ internal nodes) while the corresponding teacher output values are collected within a vector T. The optimal weights are then calculated by

$$W^{OUT} = M^{-1}T \tag{3}$$

where W^{OUT} denotes the connections to the output layer. In the third phase, the quality of the estimated weights is evaluated. To allow a direct comparison between different target time series we use the normed root mean squared error between the target signal and the network output to evaluate the quality of a certain ESN:

$$NRMSE = \sqrt{\sum_{i=1}^{I} \sum_{t=0}^{T-1} \frac{\|d_i(t) - y_i(t)\|^2}{T\sigma_i^2}}, \tag{4}$$

where I refers to the output dimension, T is the target sequence length int the exploitation phase, $y_i(t)$ is the network output at time step t in output dimension i, and σ_i^2 is the variance of the target dynamics $d_i(n)$ in the i'th dimension. There also exist online-learning procedures for training ESNs, which we do not consider here.

3 Experimental Setup

We use a simple simulator of a 2D-arm. This arm can be customized with regard to the number of segments and the length of these segments. Movement generation is carried out by first manually defining a trajectory for the endpoint. The trajectory is then reproduced by the arm, while data is recorded according to one of the coding schemes described below. The inverse kinematics are solved directly, without considering additional planning techniques.

We either recorded the position of the end-effector of the arm in target space, or the evolution of joint angles over time. Angles are measured in radiants separately for each joint. The location of the end point is recorded in pixels of a coordinate system laid over the task space. To investigate the ability of the networks to generalize the learned trajectories we did not only collect the absolute angles, or target locations, but also the differences in joint angles or end-effector coordinates between two subsequent steps. For the training we applied the raw data as well as data normalized to unity.

This data is then used for teacher-forcing. In case of joint angle data one output unit per joint is used. For the recordings of the location in target space, one output unit is used each for x- and y-coordinate, independent of the number of arm-segments. After training the ESNs, the activations of the output units in the exploitation phase are once again stored. These activations are the movement data that are then again used by the simulator to reproduce the recorded trajectory.

For our main evaluation trials, we used an arm setup with three segments, to draw different 8-shaped trajectories and recorded them with different coding schemes. Hence, the task we applied is similar to the *lazy figure eight* task described in [9]. With respect to these schemes varied crucial network parameters to identify the most effective ESN setup settings that yield the most successful learning of the trajectories. In the first set of trials, we investigated the influence of the spectral radius λ^*, the connectivity of the dynamic reservoir, as well as the initialization range of the output feedback weights W^{OFB} on the performance of the ESNs. We used seven different spectral radii, nine different connectivities, and nine different initialization ranges for the output feedback weights. Given that for every factor combination 20 networks were trained, the whole sequence required the training of 11340 networks. Ranges and step sizes of the parameter variations are displayed in the figures. As we were interested in the reproduction of the motion patterns with small networks, we kept the size of the dynamic reservoir fixed to 20 units in these runs. To estimate the influence of the different parameters, we trained 20 networks per parameter combination.

In the second set of trials another parameter setup was used. First, the connectivity of the reservoir was fixed to 0.9, as it did not show any significant influence on the results in the first trials. Instead, we varied the reservoir size from 10 to 20 units. Second, we varied the values of the initialization range of the output feedback weights W^{OFB} over a broader range. We used the same variations of the spectral radius λ^* as in the first set of trials. In this set of trials we applied seven different spectral radii, 11 reservoir sizes, and 24 initialization ranges for the output feedback weights. Again, we trained 20 networks per parameter combination, yielding a total of 30800 networks.

In the third set of trials we further investigated the influence of the reservoir size, creating networks with 25 to 100 units. The other parameters were the same as in the second run, yielding a total of 19600 networks.

Due to the large body of data and the fact that we would like to present the characteristics of the error distributions as detailed as possible, we decided to use box and whisker plots. The grayed area indicates the inter-quartile range covering 50% of the data, the outer whiskers indicate the 5% nad 95% quantile, repectively. Values out of this range but within the doubled inter-quartile range are considered as outliers (marked as open circles), values outside the doubled inter-quartile range are considered as extreme values and, if existent, are indicated with open triangles. The arithmetic mean is indicated by a black dot, whereas the smallest error value in each distribution is indicated by a gray square.

4 Results

As noted above, the coding schemes can be divided into schemes relying on joint angles and task space coordinates.

4.1 Relative Location in Target Space

We first investigated the network performance with data obtained with the normalized relative scheme, i.e. subsequent changes of the end effector position were recorded and normalized to unity. Fig. 2 displays the results of our first series of experiments. As can be deduced from the figure, the inter-quartile ranges of the different parameter setups overlap. There was not much variance due to the parameter variation and the overall performance was quite good. As it was pointed out by [8], the optimal spectral radius depends on the frequency of the desired signal. Interestingly, the best performing network was found for a spectral radius of 0.9 despite the fact that most networks generated with such a high spectral radius performed worse than networks with a smaller spectral radius.

The initialization of the output feedback weights W^{OFB} did not show a clear influence in the investigated interval and all values resulted in similarly good performance, the best results were found for an initialization interval of $[-0.08, 0.08]$. The overall effect of variations of the output feedback range was surprisingly small, therefore we decided to sample a broader interval in the next experiments.

The influence of the connectivity was completely indistinct, therefore we did not considered this parameter in the further analysis but fixed it to 0.9. Apparently, this task can be easily learned with ESNs, as even the naïve initialization could result in suitable networks.

In the second series of experiments, the variations once again failed to show more than minor effects (see Fig. 3). The error-medians remained nearly constant over the whole W^{OFB} initialization interval, but the variance increased for the extreme ranges of the initialization interval. However, the best networks were found in these ranges as well: Either with an initialization interval of $[-1.0, 1.0]$, or with an initialization interval of $[-10^{-15}, 10^{-15}]$. For this scheme varying the reservoir size had no general effect on the majority of the error-values. The effect of the variation of the error interval had no strong effect. The best performing networks were found for a reservoir size of 12 units. Concerning λ^*, a value of 0.9 led to the best results.

To sum up, the two-dimensional progression of end-effector changes in task space can be learned with quite small networks. The only parameter with more than a small effect was the spectral radius, with values above 0.9 increasing the error variance. The data pattern obtained with absolute task space coordinates was quite similar, hence we did not include the results.

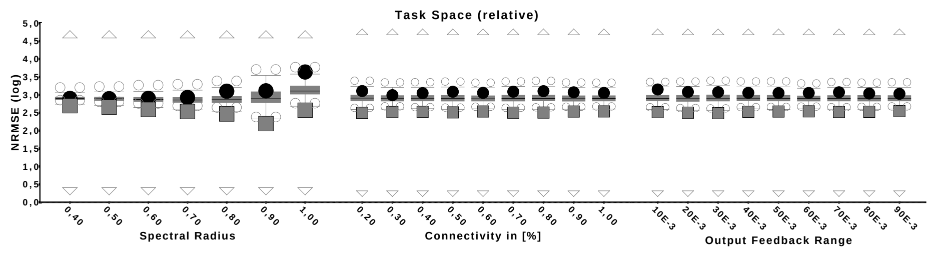

Fig. 2. Box-plots for results of parameter optimization for task space data (first experimental setup). Scaling is logarithmic. Extreme values (outside the doubled interquartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

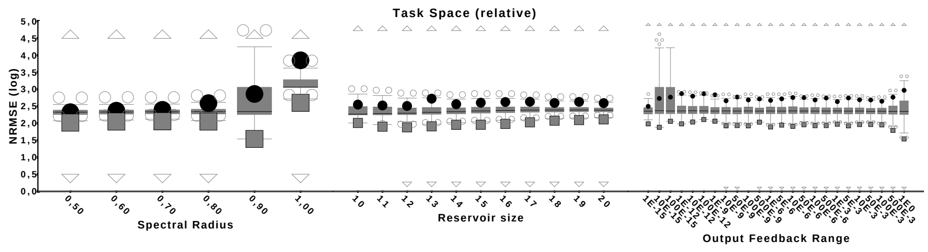

Fig. 3. Box-plots for results of parameter optimization for task space data (second experimental setup). Scaling is logarithmic. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

4.2 Joint Angles

Given the fact that the small networks we applied were able to learn a two-dimensional sequence we proceeded to the more complex task of learning a three-dimensional joint angle evolution within one dynamic reservoir. We considered two options of coding the movements in joint space. Either we recorded the absolute angles of each joint or we recorded the angular changes in consecutive time-steps. This latter relative coding proved to be problematic. Even if it was possible to learn and reproduce the time-series for the original initial arm posture, generalization was extremely poor. The reproduction of the trajectories was severely flawed when the initial arm posture was changed. Even small displacements led to major distortions of the original movement. The overall results are display in Fig. 5. Please note that these results were obtained for setups were the initial arm-posture during the reproduction was the same as during training. As noted before in this case the results are quite good, and only small effects of the parameter variations are visible. Once again a spectral radius of 0.9 turned out to produce the best results, even if the error distribution becomes broader for higher spectral radii. The variation of the initialization range of the output feedback weights only affected the broadness of the error distributions, the minimal values remained nearly unaffected. It is noteworthy that the reservoir size had only a very small effect on the overall performance — again quite small networks were able to reproduce the intended distribution. Because of the low generalization ability of networks trained with data obtained with this encoding scheme, we did not further investigate it.

More promising results were obtained with the absolute coding scheme (cf. Fig. 6). Although the output of the most successful network had a quite high NRMSE of 4.72, it produced a smooth trajectory very close to the original one (cf. Fig. 4). To reach this value, we identified several important parameters of the network. It turned out that the small reservoir size of 20 units was sufficient to reproduce the three different joint trajectories. This is in line with the results reported by [3] where it was also shown that quite small networks are able to reproduce at least three dimensional output series. For the spectral radius a value of 0.8 provided the best results. The overall effect of the spectral radius resembles the one observed in the other setups. The minimum error decreases with higher spectral radii, but the error distribution gets broader. We also varied the initialization interval of the output feedback weights W^{OFB} based on the considerations in [12], where it was pointed out that these bounds should be balanced with the interval of the target dynamics. Here, the best results were achieved with a value of 1.0 for the weight initialization interval. Again, this was a quite surprising result as the error distribution becomes much broader with higher initialization ranges, i.e. the most networks initialized with these values performed quite bad. But as the variation of the overall error range also increased the best network belonged to this sub-sample. It is noteworthy that even the smallest networks with a reservoir consisting of only 10 units were able to reproduce the intended trajectory with only slight distortions (cf. Fig. 7).

For the third experimental setup we increased the reservoir size up to 100 in several steps to examine whether we could find a maximal suitable size as reported in [12]. A major effect of this increase was a wider distribution of error values (see Fig. 8), but also a major increase in the error magnitude (see the different scaling of the y-axis in Fig. 6 and Fig. 8). Especially networks with small spectral radii and networks with large reservoirs tend to produce extreme error values. Nonetheless, a network with a reservoir consisting of 100 units also produced the lowest errors of all tested networks. For a better comparison of the minima of the different distributions, Fig. 9 displays the lower ranges of the error distributions. No clear effect of the variation of the initialization bound of the output feedback weights could be found.

To sum up, our results show that even small networks with reservoirs consisting of only 10 units are suitable to adapt to at least three-dimensional time-series. Once again, it is noteworthy that the reproduction took place in a completely self-recurrent manner, without any additional inputs. Surprisingly, the parameter variations we investigated primarily affected the broadness of the error distributions but not the extreme values. Nearly all of the investigated setups lead to the generation of at least a few suitable networks. This effect was especially prominent for the variation of the reservoir sizes, where the largest reservoir size of 100 units resulted in a nearly uniform error distribution.

5 Discussion

In this paper we described our current work on employing ESNs to learn and reproduce simple motor patterns. We mainly concentrated on finding encoding schemes to record movement data, which can be easily learned by ESNs. Compared to other approaches to learning motor primitives with reservoir computing techniques, we focused on small and simple networks similar to those applied by [3]. The first scheme we found to be viable codes the absolute angle of each joint of an arm over time. Although the quality of data reproduction depends on the parametrization of the ESN, we showed that it is generally possible to reach high reproduction accuracy with very simple networks. Additionally, this

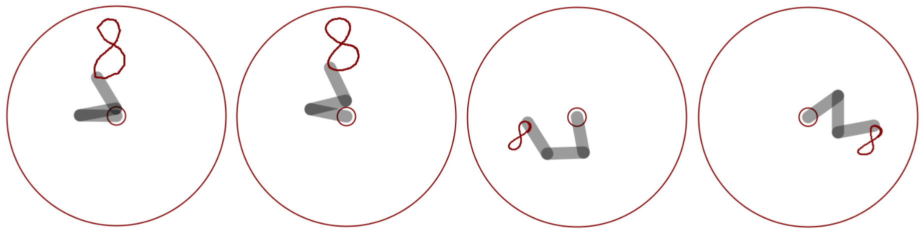

Fig. 4. Left to right: Comparison of original and reproduced trajectory coded in joint space (absolute), two examples of reproduction with the same learned target space dataset.

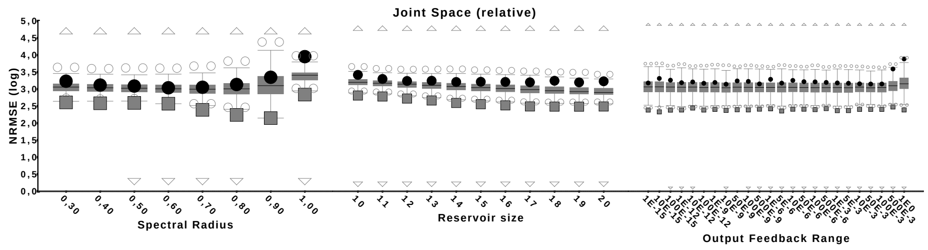

Fig. 5. Box-plots for results of parameter optimization for joint space data recorded with the relative coding scheme (second experimental setup). Scaling is logarithmic. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

scheme showed the interesting emergent effect of generating a smoothed and more symmetrical version of the original trajectory.

Scaling this scheme to 3D-movements might result in problems of the training due to the increased complexity of the data and the need for learning more time-series data simultaneously. So far we did not investigate the scaling properties of ESNs with respect to the movement data. [19] started with a network consisting of 300 neurons to learn the movement pattern of a single joint-angle, and 2400 neurons to account for a system with 22 DoF. Given our results we are optimistic to learn equally complex systems with a much smaller dynamic reservoir.

Another promising scheme encodes the movement in task space. It records the relative movement of the end-effector between consecutive time steps. This scheme has several advantages, apart from the general benefit that learning seems quite easy. First, due to the relative nature of the data, the learned movements can be simply transferred to any place in the task space, as long as it is not too close to the boundaries. Second, the scheme is completely independent of the number of arm-segments, as it only refers to the end-effector. Hence, there should be no scaling problem when extending this coding scheme to a 3D task space.

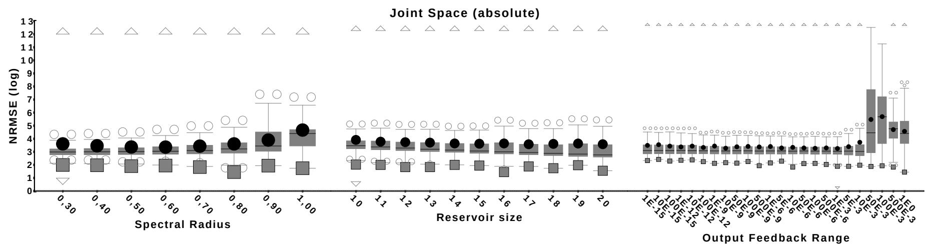

Fig. 6. Box-plots for results of parameter optimization for joint space data recorded with the absolute coding scheme (second experimental setup). Scaling is logarithmic. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

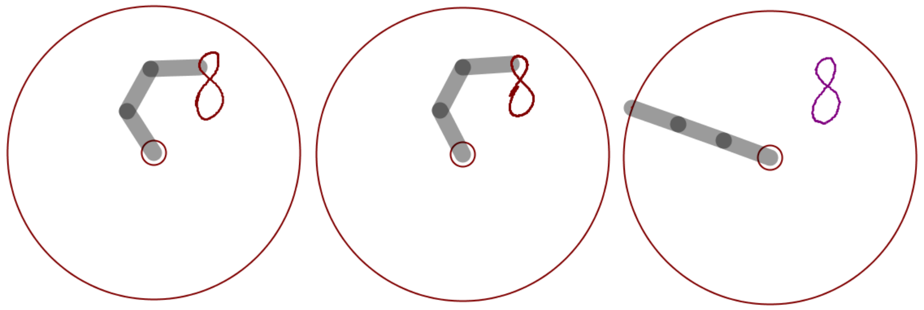

Fig. 7. The left and the middle panel display the reproduction performance of two small networks trained with absolute joint angle evolution. The rightmost panel displays the intended trajectory.

To sum up, our experiments showed that it is possible to learn simple motor pattern with quite small and simple ESNs. Even without regularization of the weight matrices (see for instance [16]), we found completely output-feedback driven networks that remained stable, but analyses regarding the long term stability are pending. As it was mentioned in the introduction, the second aspect of movement control with motor primitives, the adaptivity of movement control, is not realized yet. But as it was shown by [19] and [13], it is possible to switch between different dynamics within the same reservoir to produce different movements. This was achieved with a special input layer that served as a dynamic selection mechanism that enabled the selective activation of two different movement patterns.

On the other hand, dealing with perturbations is a much greater problem, which might not be solvable with our current approach. As it was mentioned in [1], *open-loop* approaches like the one proposed here cannot adapt very well to perturbations or delays. In [19], it was shown that reservoirs with suitable weight matrices are able to recover from perturbations and to converge back to the learned dynamic. At the moment experiments are missing that investigate if

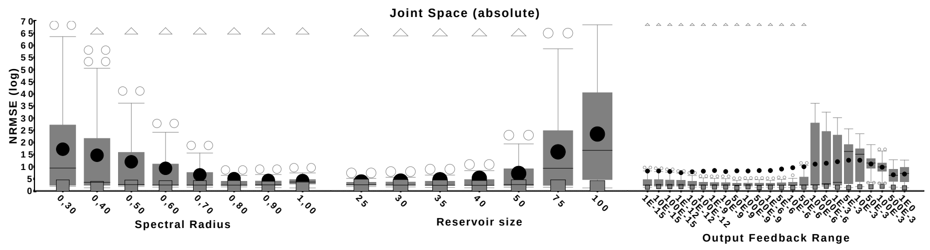

Fig. 8. Box-plots for results of parameter optimization for joint space data recorded with the absolute coding scheme (third experimental setup). Scaling is logarithmic. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

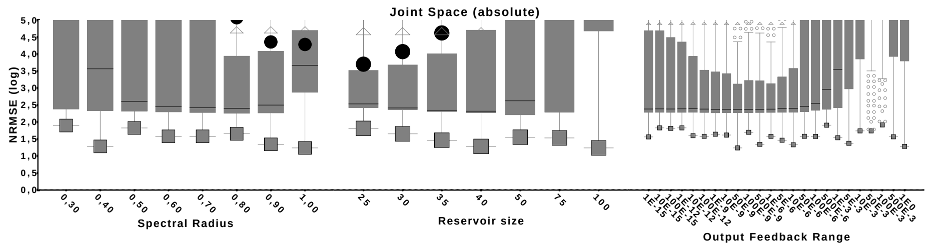

Fig. 9. Box-plots for results of parameter optimization for joint space data recorded with the absolute coding scheme (third experimental setup). Scaling is logarithmic and restricted to a range of 10^5. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

the simple networks that we trained are also able to recover from perturbations. So far we have shown that much simpler ESNs than previously thought are able to learn stable movement primitives. Our next step is to investigate if a simulated central pattern generator, like the ones proposed by [19] and [3], can be realized with small and simple ESNs.

References

1. Gribovskaya, E., Khansari Zadeh, S.M., Billard, A.: Learning Nonlinear Multivariate Dynamics of Motion in Robotic Manipulators [accepted]. International Journal of Robotics Research (2010)
2. Hart, C.B., Giszter, S.F.: A neural basis for motor primitives in the spinal cord. J. Neurosci. 30(4), 1322–1336 (2010), `http://dx.doi.org/10.1523/JNEUROSCI.5894-08.2010`
3. Hellbach, S., Eggert, J.P., Körner, E., Gross, H.M.: Time series analysis for long term prediction of human movement trajectories. In: Köppen, M., Kasabov, N., Coghill, G. (eds.) Advances in Neuro-Information Processing, pp. 567–

574. Springer-Verlag, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-03040-6_69

4. Hoerl, A.E., Kennard, R.W.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 12(1), 55–67 (1970)
5. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. Neural Networks 21(4), 642–653 (2008)
6. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on. pp. 1398–1403 (2002)
7. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Tech. Rep. GMD Report 148, German National Research Center for Information Technology (2001)
8. Jaeger, H.: Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. Tech. Rep. GMD Report 159, Fraunhofer Institute AIS (2002)
9. Jaeger, H., Lukoševičius, M., Popovice, D.: Optimization and applications of echo state networks with leaky integrator neurons. Neural Networks 20(3), 335–352 (2007)
10. Khansari-Zadeh, S.M., Billard, A.: Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. IEEE Transaction on Robotics (2011), http://lasa.epfl.ch/khansari
11. Konczak, J.: On the notion of motor primitives in humans and robots. In: Berthouze, L., Kaplan, F., Kozima, H., Yano, H., Konczak, J., Metta, G., Nadel, J., Sandini, G., Stojanov, G., Balkenius, C. (eds.) Proceedings of the Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. pp. 47–53 (2005), http://cogprints.org/4963/
12. Koryakin, D., Lohmann, J., Butz, M.V.: Balanced echo state networks. Neural Networks (2012)
13. Krause, A.F., Bläsing, B., Dürr, V., Schack, T.: Direct control of an active tactile sensor using echo state networks. In: Ritter, H., Sagerer, G., Dillmann, R., Buss, M. (eds.) Human Centered Robot Systems, Cognitive Systems Monographs, vol. 6, pp. 11–21. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10403-9_2
14. Nishimoto, R., Tani, J.: Learning to generate combinatorical action sequences utilizing the initial sensitivity of deterministic dynamical systems. Neural Networks 17(7), 925–933 (2004)
15. Pastor, P., Kalakrishnan, M., Chitta, S., Theodorou, E., Schaal, S.: Skill learning and task outcome prediction for manipulation. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on (May 2011). pp. 3828–3834 (2011), http://www-clmc.usc.edu/publications/P/pastor-ICRA2011.pdf
16. Reinhart, F., Steil, J.: Regularization and stability in reservoir networks with output feedback. Neurocomputing 90, 96–105 (2012), http://www.sciencedirect.com/science/article/pii/S0925231212001749, advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011)
17. Reinhart, R.F., Steil, J.J.: Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot icub. In: Humanoids. pp. 323–330 (2009)
18. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Networks 20, 391–403 (2007)

19. Wyffels, F., Schrauwen, B.: Design of a central pattern generator using reservoir computing for learning human motion. In: ECSIS Symp. on LAB-RS. pp. 118–122 (2009)
20. Wyffels, F., Schrauwen, B., Stroobandt, D.: Stable Output Feedback in Reservoir Computing Using Ridge Regression. In: Kurková, V., Neruda, R., Koutník, J. (eds.) Artificial Neural Networks (ICANN), Lecture Notes in Computer Science, vol. 5163, pp. 808–817. Springer Berlin / Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-87536-9_83