Design of a central pattern generator using reservoir computing for learning human motion

Francis wyffels and Benjamin Schrauwen
Electronics and Information Systems Department
Ghent University - Belgium
Email: Francis.wyffels@elis.UGent.be, Benjamin.Schrauwen@elis.UGent.be

***Abstract*— To generate coordinated periodic movements, robot locomotion demands mechanisms which are able to learn and produce stable rhythmic motion in a controllable way. Because systems based on biological central pattern generators (CPGs) can cope with these demands, these kind of systems are gaining in success. In this work we introduce a novel methodology that uses the dynamics of a randomly connected recurrent neural network for the design of CPGs. When a randomly connected recurrent neural network is excited with one or more useful signals, an output can be trained by learning an instantaneous linear mapping of the neuron states. This technique is known as reservoir computing (RC). We will show that RC has the necessary capabilities to be fruitful in designing a CPG that is able to learn human motion which is applicable for imitation learning in humanoid robots.**

I. Introduction

In all animals, vertebrates or invertebrates, neural circuits that produces rhythmic patterns of neural activity can be found [5]. These circuits are called central pattern generators and they produce rhythmic motor patterns often even in the absence of timing cues from sensory neurons or other extrinsic inputs [14].

Techniques inspired by animal central pattern generators (CPGs) are increasingly used for the generation of rhythmic signals to control locomotion of autonomous robots such as salamander robots [7], crawling humanoids [17], amphibious snake robots [3] and biped locomotion control [16]. Although their lack of a complete design methodology, motivation for their success can be found in the interesting properties that make CPGs useful for the control of locomotion as an alternative to methods based on finite-state machines, sine-generators, pre-recorded reference trajectories and other [6]. Models of CPGs have been implemented already by several kinds of techniques such as connectionist models [2], vector maps [15] and coupled oscillators [7].

Recently a novel technique for the fast training of large recurrent neural networks has been introduced independently as echo state networks [8] and liquid state machines [13], and is unified as reservoir computing (RC) [19]. In RC, the output is derived by an instantaneous, linear memory-less mapping of a large untrained dynamical system, the reservoir, which is excited with one or more inputs. The reservoir can be a randomly connected recurrent neural network which has to be rescaled so that it is operating at the edge of chaos where its processing power is greatest [12]. RC has proven its qualities in a broad range of applications such as robot localization [1], speech recognition [18] and time series generation [10].

In this paper we introduce a new methodology for CPG design based on reservoir computing (RC) which can be used for learning rhythmic motor signals. With our approach we are able to build CPG models which have interesting properties which make them suited for realistic robot locomotion. A good review of CPG models and their properties can be found in [6]. We now summarize the most important properties of a good CPG model:

- CPG models generate rhythmic patterns. The trajectories which will be performed by the joints of the robot are determined by these rhythmic patterns. To produce realistic motion patterns one can use imitation learning.
- Stable pattern generation and robustness against perturbations is necessary. The generated patterns may not deform after a time and when the state variables are perturbed the system needs to turn to its normal behavior.
- CPG models fit in a hierarchical scheme: rhythmic patterns are generated on low level, high level controllers only deal with modulation of locomotion. Thus CPG models have low dimensional inputs with which the shape of the learned signals can be controlled.
- CPG models can cope with multidimensional pattern generation. Apart from the patterns, the phase relation between the generated patterns is determined.

In what follows, we first briefly describe reservoir computing with special attention to its use as a CPG. Next, we will show the capabilities of our system to learn human motion and to exhibit rich motor skills. Therefore we trained our system with human motion capture data obtained from the CMU Graphics Lab Motion Capture Database. We perform experiments showing our system's ability to generate the learned patterns in a stable, robust and controllable way. Finally, further work will be discussed.

II. Building central pattern generators with reservoir computing

One of the main properties of a CPG is that it generates stable rhythmic patterns, for that reason we will focus this introduction on the use of RC in a generative setup. In this kind of setup, the reservoir has output feedback with additionally one or more control inputs. When using output feedback, output nodes are connected to the input of the system so

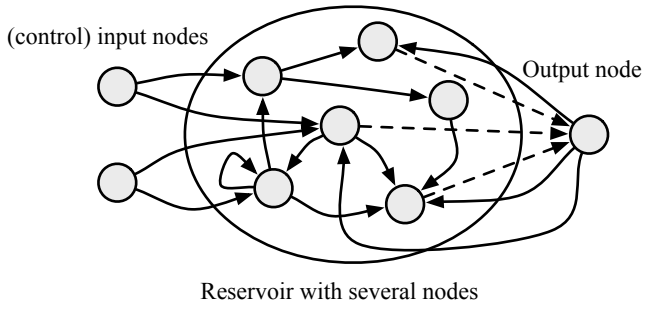

Fig. 1. Schematic overview of reservoir computing in a generative setup. This system has one ore more generative outputs with output feedback. Additional control inputs can be useful for modulation of the rhythmic signals. Only the connections directed to the output are learned (denoted by dashed lines), the other connections (in black) are fixed and initial randomly created.

that the system reacts on its outcome. A schematic overview of this is given in Figure 1. In [11] one can find evidence that feedback allows systems to generate certain movement primitives and that it can generate a suitable slow dynamics with high precision. The reservoir is composed of randomly connected sigmoid neurons. All neuron states and output are updated by use of following equations:

$$\begin{aligned}\mathbf{x}[k+1] =&(1-\lambda)\mathbf{x}[k]+\\ &\lambda f\Big(W^{\text{res}}_{\text{res}}\mathbf{x}[k]+W^{\text{res}}_{\text{inp}}\mathbf{u}[k]+W^{\text{res}}_{\text{out}}\mathbf{y}[k]+W^{\text{res}}_{\text{bias}}\Big)\\ \widehat{\mathbf{y}}[k+1] =&W^{\text{out}}_{\text{res}}\mathbf{x}[k+1]+W^{\text{out}}_{\text{bias}},\end{aligned}$$

The neuron states $\mathbf{x}[k+1]$ at time $k+1$ depends on the neuron states $\mathbf{x}[k]$ at previous time step k, an additional (control) input $\mathbf{u}[k]$, the teacher forced output $\mathbf{y}[k]$ and a bias. The system's dynamics can be effectively tuned by changing the leak-rate λ [8]. The non-linearity f represents the sigmoid function. The weights denoted by $W^{\text{res}}_{\star}$ are fixed and randomly created. For construction of the weight matrix $W^{\text{res}}_{\text{res}}$ weights are drawn from a normal distribution with zero mean and variance 1 and a large part is set to zero according the used connection fraction. The randomly connected matrix $W^{\text{res}}_{\text{res}}$ is rescaled such that the largest eigenvalue (spectral radius) is smaller than 1. This makes sure that the created system is stable and has a fading memory [9]. During training all the weight matrices denoted by $W^{\text{out}}_{\star}$ are learned using standard linear regression techniques. Because only the output weights are changed, training is extremely fast which can be an additional benefit in comparison with other methods. Additionally, reservoir computing doesn't suffer from local optima like other methods based on neural networks do. When testing the system, the teacher forced output feedback $\mathbf{y}[k]$ is replaced by the actual output $\widehat{\mathbf{y}}[k]$, this is also known as free-run mode.

In order to get good a good performance we use ridge regression to train the output weights $W^{\text{out}}_{\star}$ of our system. This keeps the outputs weights small and regularizes the trained trajectory in state-space [20] which gives our system good generalization capabilities. Using ridge regression for training introduces an additional parameter, the regularization parameter, which needs to be optimalized. Therefore we need to train and validate our system iteratively, each time using an other value (chosen in a certain range) for the regularization parameter. During validation the performance of the system is tested on a dataset different from the training set and under equal conditions as during testing. Through this work the Normalized Mean Squared Error (NMSE) is used for evaluating the performance on the validation set. Notice that when performing generation tasks the training set and validation set can be the same because the goal is to generate the learned samples. When an optimal regularization parameter is found the system is retrained using the optimal value. Finally the system can be tested on an unseen test set.

III. Modeling and control of human motion

A. Data gathering and preprocessing

The data we used in this article was obtained from the CMU Graphics Lab Motion Capture Database (see acknowledgment). The data consists of the 3D joint angle evolution for 30 markers. The data was recorded at 120 Hz and we used it without any sub-sampling. Because we are only interested in walking and running motion of humans we only used data of subject 35 from the CMU database, this subject had most walking and running example sets available. In order to avoid saturation of the sigmoid neurons, data was normalized by subtracting the mean and dividing by its variance.

B. Generation of learned rhythmic motion patterns

What makes a good candidate of RC for designing CPGs is that much of the desired properties of CPGs can be realized by use of RC. A CPG produces rhythmic patterns, in [8] the ability to learn and generate time series has been shown. Both stable output feedback and robustness can be seen in generation tasks and is discussed in previous work [20], [4]. In the top graph of Figure 2 one can see that our system is able to generate a learned periodical signal in form of an angle joint evolution of the left femur (legs) from subject 35 in the CMU database. Only one set containing 358 samples was used during training. In the top graph of Figure 2 the reservoir output is shown in black while the desired outcome, constructed by concatenating the training set multiple times, is denoted by a gray dashed line.

The middle graph of Figure 2 illustrates that our system is able to generate the learned signal over longer time spans. Here evolution of the joint angle is shown in black after more than $100,000$ samples have autonomously been generated by the system. To achieve this stability it was necessary to use ridge regression [21]. We used a small validation set of 500 samples (angle joint evolution of the left femur) during which the reservoir ran in free-run mode and was perturbed with noise sampled from a Gaussian distribution with zero mean and variance 0.001 in order to validate for dynamical stability. By adding noise during validation we are able to control the smoothness of the learned signal: more noise during validation simulates the effect of a long free-run trial and will cause more stability but less accuracy (but high accuracy of learned

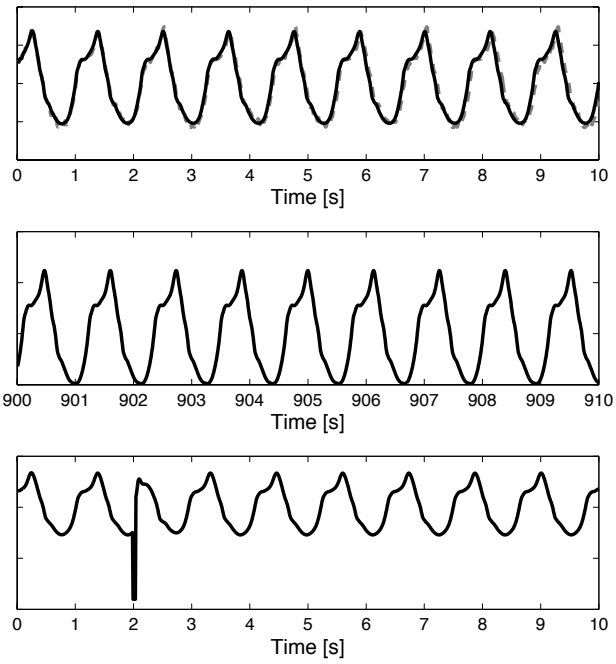

Fig. 2. This figure shows the ability to stably generate rhythmic signals after learning. In the top graph the joint angle evolution of the left femur is shown in gray dashed lines. The system's output during free-run is shown in black. We see little difference between the reservoir output and the teacher signal. In the middle graph we see a continuation of the system's output after 900 seconds of running freely. This illustrates the stability capabilities of our system. In the bottom graph we see the system's capability to cope with a large perturbation: after 2 seconds of generation the evolution of the joint angle of the left femur, the output was forced to a random outlier value during 50 ms. We see that the network recovers well after such a perturbation.

signals is not needed in robot locomotion). The regularization parameter was searched in the range $10^{[1:-0.1:-8]}$.

Apart from the ability to generate stable rhythmic motor patterns it's also necessary for a CPG to recover from perturbations. In the bottom plot of Figure 2 this property is illustrated. After 2 seconds the reservoir output was clamped to an outlier value for 50 ms. We see that the system recovers well after removing this perturbation. In order to have such robustness we had to use ridge regression and to optimize the regularization parameter. This was done in the same way as described earlier.

For all of the previously described experiments we used a reservoir of 300 randomly connected (connectivity of 50%) leaky integrator neurons [8] with a leak rate of 0.3. The spectral radius was 1 and each neuron had an auxiliary input with a bias sampled from the uniform interval $[0; 0.1]$. Only output feedback was used with a scaling factor sampled from a binomial distribution with equal probability to have -0.05 or 0.05. These settings were chosen arbitrarily and none of them is critical. In order to generate the plots in Figure 2 the reservoir states were initialized before free-run mode starts. This was done by updating the reservoir state with the update equation given in equation 1 which uses the teacher forced output as for feedback.

In order to reduce the dimensionality of the control problem, the CPG needs to modulatable such that high level controllers only produce high level control signals [6]. One of the basic signal properties that one wants to modulate is the frequency. The black curve in Figure 3 shows that we are able to modulate the frequency of the learned signal. This property of RC has been previously illustrated on a sine wave in [10]. For modulation of a learned rhythmic pattern from mocap data, we constructed a training set by concatenating the joint evolution of the left femur from subject 35 until we had a large dataset and subsample it in function of the values of an additional control input. This input signal varied between 1 to 6 during training in a smooth way causing a maximal increase of frequency by six times its original frequency. No abrupt changes in frequency took place. During this experiment the topology of the reservoir was similar to the one described before with only one change: the described control input was connected to 25% of the reservoir neurons. This was enough to have the reservoir influenced by the input. Total training size was $20,000$ samples. In order to have sufficient stability while changing the input, it was necessary to use ridge regression. The regularization parameter was optimized during validation as we described earlier. The validation set was constructed the same way as the training set and contained $1,000$ samples. For testing, a different input signal was used to modulate the learned signal. Both, the system's output and the control input can be seen in Figure 3 and are represented by a solid black curve and a dashed gray line respectively.

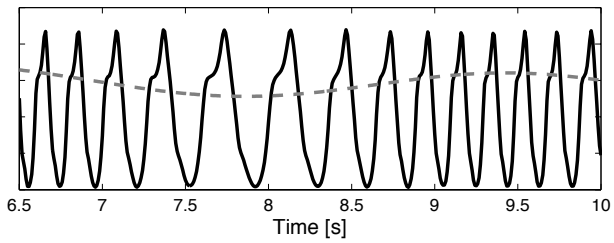

Fig. 3. This figure illustrates modulation capability of our system. The black curve shows the evolution of the learned rhythmic signal which we are able to modulate in frequency during free-run mode. The corresponding control input, illustrated with a gray curve, was switched smoothly from a high value to a low value and back to a high value again causing a smooth change in frequency from high to low, to high again.

If we want that robots show a rich motion repertoire, the CPG has to be able to deal with multiple types of gait which can be controlled by low dimensional control signals. The bottom plot of Figure 4 shows that we are able to switch to another learned gait. Because two patterns had to be learned, we created a larger reservoir containing 600 neurons using the previously described parameter settings. We added two to additional control inputs, u_{walk} and u_{run}, which were each randomly connected to 50% of the neurons. The scaling of these control inputs was drawn from a binomial distribution with equal probability to have -0.2 or 0.2. During training the system was trained with $1,000$ concatenated samples of evolution of the left femur joint angle during walking (see left top of Figure 4 for an example subset) while inputs u_{walk}

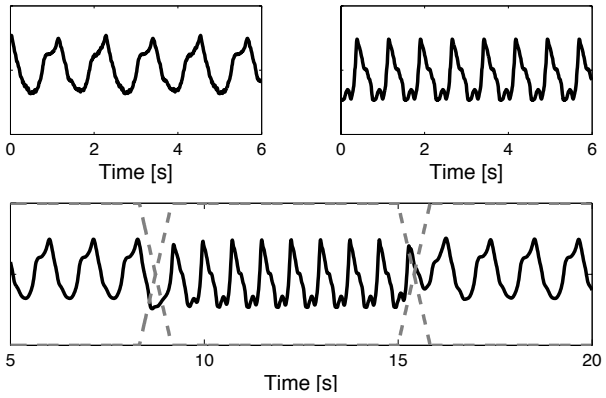

Fig. 4. Shape transitions between two learned motion patterns is presented here. The top left plot shows the evolution of the left femur joint angle while walking, the top right plot shows joint angle evolution while running. These two examples were used for training. During training the reservoir had two additional inputs denoting whether the reservoir was excited with a walking or running set. The bottom graph shows in black the evolution of the systems output during free-run mode. After approximately 8 seconds, the two control inputs (dashed gray lines) were switched which results in a transition from walking to running. The inputs were switched again after 15 seconds resulting in a transition from running to walking. It is important to notice the smoothness of the transitions when inputs are changed.

and u_{run} were set to 1 and 0 respectively. The system was also trained with a set of $1,000$ concatenated samples of joint angle evolution of left femur for running gait (an example subset of this can be seen in the top right plot of Figure 4) with input u_{run} set to 1 and input u_{walk} set to 0. Validation took place by letting our system run freely while input u_{run} was set to 1 causing the system generate the joint angle evolution for a running gait. In order to get stable generation of the desired patterns, the reservoir was perturbed with noise sampled from a Gaussian distribution with zero mean and variance 0.001 during validation. The validation set was equal to the training set. After validation, our system was retrained using the optimal regularization parameter and left running freely. After approximately 8 seconds, the control inputs u_{walk} and u_{run} were switched smoothly from 1 to 0 and vica verse causing a smooth transition from walking to running which can be seen in black at the bottom graph of Figure 4, the control inputs are represented by a dashed gray line. After 15 seconds, the control inputs were switched again. Although we gave our system no examples of transition between different gaits during training, we are able to generate smooth transitions which are necessary in robot locomotion control. This is partly due to the validation process but also because of the two inputs with which we control the chosen gait. The way we choose these control inputs is important for the outcome we have. Abrupt change of these control inputs will result in large amplitude spikes of the system's outcome which is not desired.

C. Generation of multiple patterns

In order to be suitable for robot locomotion the CPG has to be able to generate motor signals for each joint of the robot. It is crucial that all these signals are coupled, otherwise desychronized movements of arms and legs will occur while

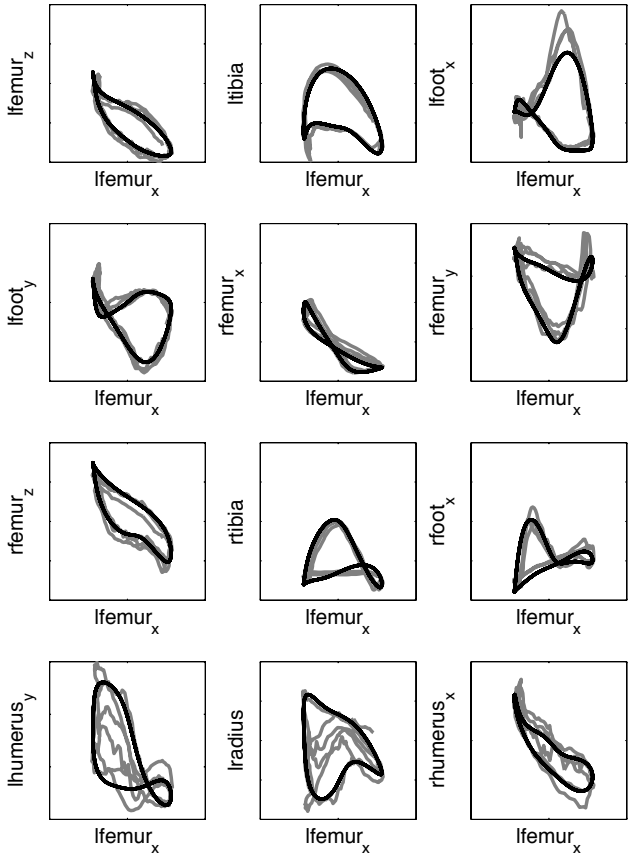

Fig. 5. The twelve phase portraits show the evolution of several joint angles in function of the left femur joint angle evolution. In light gray, the attractor shows the phase coupling of the original data set gain from the CMU database. Some plots are distorted due to noise in the original sets. The attractors in black are derived from the free-run output of our system during 10 seconds after running more than 500 seconds. One can see tight phase coupling causing stable attractor generation.

walking, or even no walking will be realized when for example all leg joints are in wrong phase. With our system based on RC it is easy to learn multidimensional rhythmic patterns. And because all these patterns are derived from the same dynamical system (the reservoir), also the phase relation between the desired patterns will be learned without any difficulty.

We illustrate the ability to learn multidimensional patterns and their coupling by training our system with mocap data set 2 (walking) from subject 35. The degree of freedom (DOF) of the training set was reduced from 62 to 22: only arms and legs movement which can be mapped directly to the joints of the Fujitsu Hoap-2 robot were considered. Because of the amount of DOF we created a large reservoir consisting of $2,000$ neurons. Apart from the reservoir size and the increase of the number of outputs which were each connected to 65% of the neurons with a scaling factor sampled from a binomial distribution with equal probability to have -0.11 or 0.11, we used the same parameter settings as described in the previous section.

Because stable generation of the learned multidimensional patterns is desired, we used ridge regression and optimized

the regularization parameter during validation the same way as described in the other experiments. Again, during validation, the reservoir states were perturbed with Gaussian noise with zero mean and a variance of 0.001. After validation, the system ran in free-run mode for more than $60,000$ samples. In order to illustrate the phase locking between the 22 outputs of our system, the outputs are shown in a phase portrait were several outputs (leg and arm joints evolution) are plotted against the outcome for the left femur. In the nine plots in Figure 5 the phase portraits constructed from the original (mocap) data are shown in gray, the outcome of our system during 10 seconds after running for more than 500 samples freely is plotted in black. We see that phase relation remains intact. Notice how the phase portraits derived from the original dataset are deformed by noise caused by the noisy mocap data. Also, the three phase portraits derived from the original dataset at the bottom in Figure 5 are highly disturbed because arm joints are plotted against the left femur which is in the leg and one can imagine that arms are not moving in a strict rhythm as the legs do while walking. One can see that our system's outcome doesn't show these disturbances what illustrates that our system is able to learn only the desired rhythmic patterns and is not influenced by the noise in the mocap data.

IV. Conclusions and future work

In this work we proposed a new methodology for the design of central pattern generators based on reservoir computing. We have shown that our system is capable of learning rhythmic signal by example very efficiently. The resulting system is able to stably generate the learned signals and to be robust against perturbations. We showed the importance of using good regularization by means of ridge regression, a method which allows us to control the smoothness of the learned signals.

By adding one or more control inputs we can tune the frequency and shape of the learned patterns which reduces the dimensionality of the control problem such that higher level controllers only need to produce high level control signals and not the rhythmic patterns themselves.

By experiment, we showed also that it becomes easy to learn multiple signals and their phase relation which is of great importance for controlling multi-DOF robots. The experiments with multi-dimensional signals showed that reservoir size was an important parameter. The scalability of our system will be investigated in future work.

In future work we plan to use our system for stable walking of a humanoid robot by tuning the low dimensional control inputs of the CPG and to add an unsupervised learning to extract different types of motion.

Acknowledgment

The data used in this work was obtained from http://mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. For Matlab playback of motion and generation of videos, we have used Neil Lawrence's motion capture toolbox which can be found on http://www.cs.man.ac.uk/∼neill/mocap/. This research is partially funded by FWO Flanders project G.0088.09 and the Photonics@be Interuniversity Attraction Poles program (IAP 6/10), initiated by the Belgian State, Prime Minister's Services, Science Policy Office.

References

[1] Eric Aislan Antonelo, Benjamin Schrauwen, and Dirk Stroobandt. Event detection and localization for small mobile robots using reservoir computing. *Neural Networks*, 21:862–871, 2008.

[2] P. Arena. The central pattern generator: a paradigm for artificial locomotion. *Soft Computing*, 4:251–266, 2000.

[3] A. Crespi and A.J. Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *IEEE Transactions on Robotics*, 24:75–87, 2008.

[4] Xavier Dutoit, Benjamin Schrauwen, Jan Van Campenhout, Dirk Stroobandt, Hendrik Van Brussel, and Marnix Nuttin. Pruning and regularization in reservoir computing. *Neurocomputing*, 2009. accepted.

[5] Sten Grillner. Biological pattern generation: the cellular and computational logic of networks in motion. *Neuron*, 52:751–766, 2006.

[6] A.J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21:642–653, 2008.

[7] A.J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420, October 2007.

[8] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.

[9] Herbert Jaeger. Short term memory in echo state networks. Technical Report GMD Report 152, German National Research Center for Information Technology, 2001.

[10] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. *Science*, 308:78–80, April 2 2004.

[11] Prashant Joshi and Wolfgang Maass. Movement generation with circuits of spiking neurons. *Neural Computation*, 17:1715–1738, 2005.

[12] R. A. Legenstein and W. Maass. Edge of chaos and prediction of computational performance for neural microcircuit models. *Neural Networks*, pages 323–333, 2007.

[13] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[14] Eve Marder, Dirk Bucher, David J. Schulz, and Adam L. Taylor. Invertebrate central pattern generation moves along. *Current Biology*, 15:R685–R699, 2005.

[15] M. Okada, K. Tatani, and Y. Nakamura. Polynomial design of the nonlinear dynamics for the brainlike information processing of whole body motion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[16] L. Righetti and A.J. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 1585–1590, 2006.

[17] Ludovic Righetti and Auke J. Ijspeert. Design methodologies for central pattern generators: an application to crawling humanoids. In *Proceedings of Robotics: Science and Systems*, pages 191–198, 2006.

[18] Benjamin Schrauwen, Michiel D'Haene, David Verstraeten, and Jan Van Campenhout. Compact hardware liquid state machines on fpga for real-time speech recognition. *Neural Networks*, 21:511–523, 2008.

[19] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20:391–403, 2007.

[20] F. wyffels, B. Schrauwen, and D. Stroobandt. Stable output feedback in reservoir computing using ridge regression. In *Proceedings of the International Conference on Analog Neural Networks (ICANN)*, 2008.

[21] F. wyffels, B. Schrauwen, D. Verstraeten, and D. Stroobandt. Band-pass reservoir computing. In *Proceedings of the International Joint Conference on Neural Networks*, 2008.