Physical Reservoir Computing in Robotics

Helmut Hauser

Abstract In recent years, there has been an increasing interest in using the concept of physical reservoir computing in robotics. The idea is to employ the robot's body and its dynamics as a computational resource. On one hand, this has been driven by the introduction of mathematical frameworks showing how complex mechanical structures can be used to build reservoirs. On the other hand, with the recent advances in smart materials, novel additive manufacturing techniques, and the corresponding rise of soft robotics, a new and much richer set of tools for designing and building robots is now available. Despite the increased interest, however, there is still a wide range of unanswered research questions and a rich area of under-explored applications. We will discuss the current state of the art, the implications of using robot bodies as reservoirs, and the great potential and future directions of physical reservoir computing in robotics.

1 Introduction

Reservoir computing as a term has been coined by Schrauwen et al. (2007). Their fundamental insight was that the underlying principles of Echo State Networks (ESN) (Jaeger and Haas 2004) and Liquid State Machines (LSM) (Maass et al. 2002), which have been developed independently from each other, are the same. Both use complex, nonlinear dynamical systems (referred to as *reservoirs*) as a computational resource. While LSMs, inspired by the structure of the brain, are using differential equations describing neurons, ESNs are employing simple, but abstract nodes in form of nonlinear differential equations to achieve the same results. Both use simple, dynamic building blocks to construct a high-dimensional, stable, but nonlinear dynamical system, i.e. the *reservoir*.

H. Hauser (✉)
Department of Engineering Mathematics, University of Bristol, Woodland Road, Bristol BS8 1UB, UK
e-mail: helmut.hauser@bristol.ac.uk

Bristol Robotics Lab, Coldharbour Ln, Bristol BS16 1QY, UK

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_8

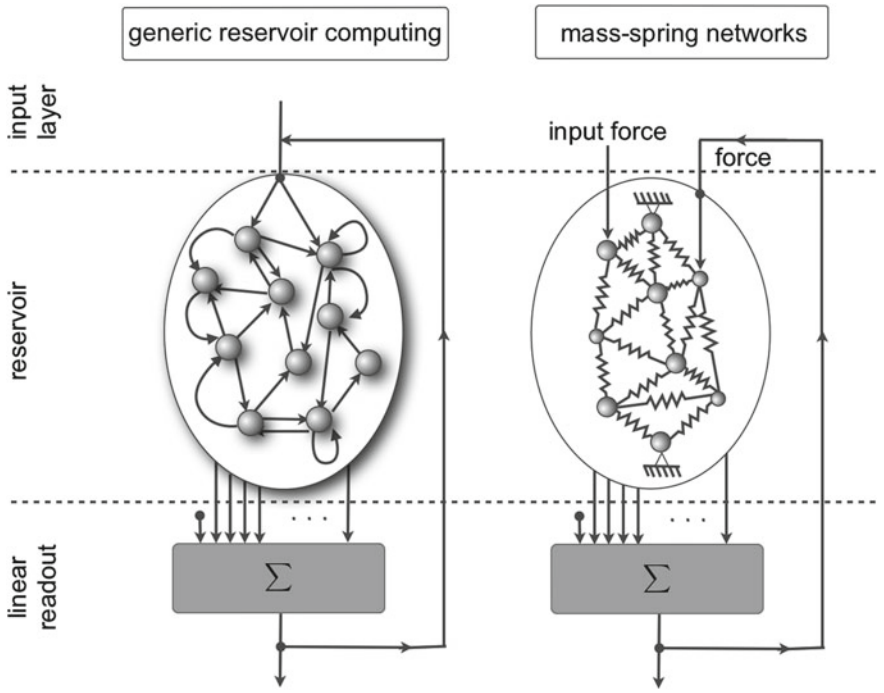

Fig. 1 Comparison of conventional reservoir computing (example is with ESNs nodes) with physical reservoir computing using mass–spring–damper systems as suggested by Hauser et al. (2011). Figure adapted from Hauser et al. (2014)

The role of a reservoir can be described as a *kernel* in the machine learning sense for support vector machines, see, e.g. Vapnik (1998). Low-dimensional input signals (typically in form of time series) are fed into the reservoir and projected nonlinearly into its high-dimensional state space. It is well established that this boosts the computational power of anything that comes after it. For a discussion in more detail in the context of reservoir computing, we refer to Hauser et al. (2011, 2012, 2014). In addition, since a reservoir is built up by smaller dynamical systems (i.e. differential equations), it also integrates information over time. Consequently, loosely speaking, besides the kernel property, the reservoir exhibits also some form of (fading) memory. This leads us to an interesting consequence. Since the reservoir is already able to integrate information and carry out nonlinear mappings, it suffices to add a simple linear, static readout to obtain a powerful computational device. We only have to find a set of static, linear weights to combine the signals from the high-dimensional state space of the reservoir to achieve the desired output (compare Fig. 1).

The resulting setup, i.e. reservoir plus linear readout, is a surprisingly powerful computational device. It can be used to approximate in principle any computation that

can be represented by a Volterra series. In practice, this means we can emulate any smooth (i.e. sufficiently derivable) dynamical system with one equilibrium point.[1]

The remarkable part of this setup however is how we achieve learning. We don't change any parameters of the reservoir (typically it is randomly initialized within some parameter bounds), but we only have to adapt/find the optimal readout weights. Since they are a set of fixed (static) values used to linearly combine the states of the reservoir, we can employ simple linear regression. Note that this means that we can learn to emulate nonlinear dynamical systems with such a setup by using simple, linear regression. One could say the integration and nonlinear combination of information (which are both needed in a nonlinear, dynamical system) are outsourced to the reservoir or the reservoir is exploited as a computational resource.

From a machine learning perspective, reservoir computing is an alternative approach to Recurrent Neural Networks (RNN), which are typically employed to approximate dynamical systems. However, it is well known that the learning process for RNNs, i.e. to find the optimal connectivity weights, is tedious. There exist various approaches, e.g. Backpropagation Through Time (BPTT) and its variations, but they are all prone to get stuck in a local minimum or are known to be slow in their convergence. On the other hand, in reservoir computing, we only have to find a linear, static set of output weights by employing linear regression, which by definition finds always the globally optimal solution[2] and is very fast.

The step from *standard* reservoir computing as described above to *physical* reservoir computing is straightforward when we consider which properties a reservoir has to have to be computationally useful. As pointed out before, the reservoir has (i) the role of mapping nonlinearly low-dimensional inputs into a higher state space and (ii) to integrate information over time. Interestingly, these properties are rather abstract and, as a consequence, a wide range of dynamical systems can be used as a reservoir. They only have to exhibit nonlinear dynamics with a fading memory property (i.e. being exponentially stable with one equilibrium point) and a high-dimensional state space.

For example, in the case of LSMs, a network of spiking neuron models is used to construct such a system, while in ESNs, randomly connected structures of simple nonlinear differential equations are employed. This means any nonlinear, exponentially stable (with one equilibrium point), high-dimensional dynamical system can be potentially used as reservoirs, which naturally includes also real physical systems.

The earliest work demonstrating that this is possible was done by Fernando and Sojakka (2003). They used the water surface in a bucket as a reservoir to carry out vowel classification. The input was sound waves exciting the water and the readout was carried out through the pixelated version of video recordings of the water surface. Since then, a wide range of nonlinear physical systems have been proposed and shown to be useful as reservoirs. Examples include nonlinear effects in lasers

[1] The relation between Volterra series and dynamical systems is discussed in Boyd (1985).

[2] Linear regression minimizes the quadratic error against the target. Since the error landscape is quadratic, there is only one (global) minimum. Note that this doesn't necessarily mean that reservoir computing setups are always performing better than RNN networks.

(Smerieri et al. 2012), body dynamics of soft robots (Nakajima et al. 2018a), and even quantum-mechanical effects (Fujii and Nakajima 2017). Here, in this chapter, we will concentrate on applications in robots.

In the next section, we will re-introduce the underlying theoretical models that have contributed to a better understanding of how the dynamics of robot bodies can be used as reservoirs. In Sect. 3, we discuss the state of the art and give an overview of existing work in this area, which is followed by a critical assessment of the advantages (Sect. 4) and limitations (Sect. 5) of the approach. In Sect. 6, we will highlight the close connection of physical reservoir computing and the recently emerged research field of soft robotics. Finally, in Sect. 7, we discuss the future of physical reservoir computing in robotics and the great potential it holds.

2 Theoretical Models

Physical reservoir computing in robotics has gained a lot of traction recently due to the introduction of mathematical models proving that certain mechanical structures can serve as reservoirs. Particularly, two models proposed by Hauser et al. (2011, 2012) were able to demonstrate that complex, mechanical architectures can serve as powerful reservoirs. They are composed of nonlinear mass–spring–damper systems which are connected to form a mechanical network (see Fig. 1 on the right). The motivation to use this approach was that these structures serve as good models to describe the body of biological systems and soft-bodied robots. Both exhibit rather complex and nonlinear dynamics that can be exploited in a physical reservoir computing setup.[3]

The proposed mathematical frameworks for physical reservoir computing based on mass–spring–damper systems are an extension of the original work by Maass et al. (2002, 2007) which show that LSMs can indeed carry out computations. While they prove the concept using models of spiking neurons and neural connection to construct a computationally useful structure (i.e. a reservoir), Hauser et al. demonstrated the same is possible with mechanical structures.

As with the original work, there has been a distinction between the two main reservoir computing setups. The so-called *feedforward setup*[4] maps an input directly (through the reservoir and readout) to the desired output, while the *feedback setup* also includes feedback loop(s) from the output back into the reservoir. Note that Fig. 1 shows the setup with explicit feedback loops. We will discuss both approaches now in more detail.

[3] Note that this also means that one could use biological bodies directly as reservoirs as well. One just has to find a way to read out the (at least partial) dynamic state of the system.

[4] Note that the term *feedforward* highlights here the fact that there are no explicit feedback loops from the output back into the reservoir. However, the reservoir naturally will still have stable feedback loops inside the reservoir to achieve the fading memory that is a prerequisite for reservoir computing.

2.1 Feedforward Setup

As mentioned before, the mathematical proof for the feedforward setup has been given by Hauser et al. (2011) and is based on the work by Maass et al. (2002). Both are based on a seminal paper by Boyd and Chua (1985), which explored the idea to emulate arbitrary Volterra series. Note that a Volterra series is an elegant way to approximate nonlinear, differential equations that have one equilibrium point.[5]

Boyd and Chua showed that any Volterra series can be approximated by a two-stage process, where the first stage has to be dynamic (include memory), but can be linear, and a second stage that has to be nonlinear, but can be static. The remarkable part is that these two stages can be implemented in any way as long as they fulfil certain properties. Specifically, the first stage has to exhibit the property of fading memory and there has to exist a pool of subsystems that is rich enough so that we can guarantee that we can always find two subsystems to separate any possible input signals (through filtering).[6] The second stage needs to be a universal function approximator, i.e. a system that is able to approximate any smooth enough nonlinear function with arbitrary precision. As one can see, these properties are very general and there are a lot of potential candidates that can embody them. Boyd and Chua gave a number of examples of possible implementations in their original work, mostly from the field of electrical circuits.

To appreciate the importance of their result, it's crucial to understand the concept of a Volterra series. Loosely speaking,[7] this could be understood as a Taylor series expansion that includes time. Instead of only approximating smooth, nonlinear functions, a Volterra series can approximate nonlinear dynamical systems. In the context of reservoir computing, especially physical reservoir computing, this gives us an elegant way to describe analogue computation. Instead of having to rely on a digital interpretation and the Turing machine concept, dynamical systems are a powerful way to describe analogue computational functions in the context of robotics. They can be used to express, for example, complex sensor functions (e.g. filtering) as well nonlinear controllers.

The only[8] limitation of using Volterra series as descriptors is they can only approximate nonlinear dynamical systems that have only one exponentially stable equilibrium point (or approximate only in the neighbourhood of an equilibrium point). However, we have to emphasize that this is still a very powerful and rich set of possible computations that can be represented. It includes nonlinear mappings that use memory. This means the output will not simply depend on the current input (that would be a simple function and could be approximated with a standard Arti-

[5] Or they approximate only the close neighbourhood of one equilibrium point, i.e. local approximation. Note that this is different from linear approximation approaches, e.g. expressed through a Jacobian matrix, since a Volterra series is still capturing nonlinearities.

[6] For example, we can achieve separation through different integration time constants.

[7] For more rigorous and mathematically sound descriptions, please refer to Boyd and Chua (1985).

[8] Note that there are more mathematical details, but for the sake of simplicity, we discuss only the main points. For a more detailed discussion, we refer to Boyd (1985) and Boyd and Chua (1985).

ficial Neural Network), but also on the history of input values, which would need a Recurrent Neural Network (RNN) to approximate it.

The question is now how can we use morphological structures, specifically mass–spring–damper systems, to approximate the Volterra series. The answer can be directly derived from Boyd and Chua's work. They laid out clearly the required properties for potential building blocks to construct the first stage of the process. They have to be (exponentially) stable and be able to separate signals as discussed before. The proof for linear mass–spring–damper systems to have these properties is straightforward (for more details, see Hauser et al. 2011).

For the second stage, the requirement is that it should be made up of a universal function approximator. We can use ANNs, which have exactly this property (see Hornik et al. 1989 for proof). Note that this two-stage setup does not yet resemble a standard reservoir setup, since the readout is still nonlinear and the first stage consists of a set of parallel, independent systems. Nevertheless, one can already say that at least the "memory part" (integration of information) is outsourced to stage one, i.e. the morphological part. However, a set of parallel mass–spring–damper system doesn't look very much like a common robot design,[9] nor does it describe well a biological body.

To get to a reservoir computing setup, we have to push Boyd and Chua's approach to the extreme by considering to outsource nonlinearity also to stage one, leaving a simplified readout that can be static and linear. This leads to networks like the one shown on the right side of Fig. 1. Note that this push "breaks" Boyd and Chua's proof and, so far, there is no mathematical proof for the full network structure (also not in the original work by Maass et al.). However, the step from the two-stage process to the full structure is reasonable and simulation results (as well real-world examples in the case of mechanical structures) demonstrate that it works.

As previously pointed out, the only limitation is that the feedforward setup is restricted to systems that can be represented by a Volterra series. Since this is already a very rich class of possible mappings, the question arises, which systems that might be interesting in robotics don't fall into this category? This will be answered in the next section on feedback setups.

2.2 Feedback Setup

Boyd already hinted at this limitation in his Ph.D. thesis (Boyd 1985). He made a link between the Volterra series and nonlinear dynamical systems. He laid out that the Volterra series can only approximate systems that have only one exponentially stable equilibrium point. Keeping the discussion in the realm of dynamical systems, it is then quite straightforward which dynamical systems fall outside of this definition, but would be still interesting in the context of robotics. First, we can think about multiple

[9] Note that, interestingly, Shim and Husbands (2007) suggested a setup which looks very similar and is used for a feathered flyer and which predates Hauser et al.'s work.

equilibrium points. Depending on the situation/task, different endpoints might be of interest. Another type of dynamical systems that go beyond exponentially stable ones are (nonlinear) limit cycles. These are particularly interesting in the context of robotics as limit cycles are a way to encode repetitive movements, e.g. in locomotion. In addition, other nonlinear effects like bifurcation go beyond a Volterra description as well. This could be potentially helpful if we want to control a qualitative change in the behaviour. For example, the robot could switch smoothly from a locomotion movement of its front leg (i.e. limit cycle) to a reaching movement (i.e. equilibrium point)—all described by one set of differential equations.

Interestingly, it is quite simple to overcome the limitations of the feedforward setup in reservoir computing. The solution is to counteract the energy loss in the reservoir. We simply have to add (linear) feedback loops[10] from the output of the setup. Again, in standard reservoir computing, these signals are abstract streams of information that, through the feedback, will change the state of the reservoir. However, in physical reservoir computing, the feedback is a real physical value and provides energy to the mechanical body, e.g. in form of forces.

Although it seems adding a feedback loop is a rather small change compared to the bigger picture, the previous mathematical models based on Boyd and Chua's work are not applicable anymore. Fortunately, there exists a framework that can deal with feedbacks, i.e. non-fading memory tasks. Maass et al. (2007) employed the theory of feedback linearization from control theory to prove that neural models can be modified via static, but nonlinear feedback and readouts to approximate almost any smooth dynamical system. The limitations are only the smoothness and the degree of freedoms, i.e. a two-dimensional system as a building block can only approximate other two-dimensional systems.[11] One simplified way is to look at the standard model to describe an n-dimensional nonlinear differential system, i.e. $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u$ with $\mathbf{x} \in \mathbb{R}^n$. This means we have n integrators which depend on the previous states through the nonlinear mapping $f(\mathbf{x})$ and on the input through $g(\mathbf{x})$. Feedback linearization is able to overwrite both functions, i.e. $f(\mathbf{x})$ and $g(\mathbf{x})$ through applying, loosely speaking, the inverse effect.[12] In addition, we can add to the feedback a nonlinearity that "overwrites" the linearized system with new, desired dynamics in order to emulate the desired target computation, i.e. dynamic input–output mapping. Note that this is similar to finding a controller for a linear system by applying a pole-placement approach. We want the overall system to behave in a certain way (which is reflected by desired eigenvalues of the system matrix) and we try to find the right feedback controller to achieve that, for example, with Ackermann's method.

[10] There can be one or multiple feedback loops.

[11] Note that one can argue that combing multiple such systems would then again allow us to approximate systems of higher order as well, see Maass et al. (2007).

[12] Note that not every nonlinear system is feedback linearizable. However, there is a formal process to check for that by applying Lie derivatives. Note that this is equivalent to constructing the controllability matrix $\mathbb{C} = [B, BA, BA^2, \ldots, BA^n]$ in linear systems. We refer to the reader to Slotine and Lohmiller (2001) for an excellent introduction. For a more in-depth discussion, we refer to Isidori (2001).

Without going into details, the mathematical proof employed in Hauser et al. (2012) is demonstrating that certain types of nonlinear mass–spring–damper systems are feedback linearizable, i.e. can be used as a basic template that can be overwritten as new desired dynamics.

However, the proof presented by Hauser et al. (as well in the original work by Maass et al.) is only for one single system, i.e. one mass–spring–damper system, and it needs static, but still nonlinear feedbacks and readouts. The mathematical proof for more complex systems is nontrivial due to the complexity. Nevertheless, as before with the feedforward setup, the step to full network structures with linear readouts and linear feedbacks (as in Fig. 1) is reasonable. This is supported by numerous simulation results as well as real-world examples that show the usability of the approach.

While the models for the feedforward setup give us some (however) general guidelines, the models for the feedback setup, unfortunately, are not very insightful with respect to how to build better reservoirs. Maybe the only point is that such systems can be rather small in size to be useful. This has been shown and discussed with simulation examples in Hauser et al. (2012).

In the context of physical reservoir computing in robotics, however, we are very interested in understanding how to design and build better reservoirs. Therefore, the following section discusses how abstract concepts in reservoir computing are mapped onto real physical interpretations in physical reservoir computing setups.

2.3 Connecting Theoretical Models to the Real World

While conventional reservoir computing does not care about the physical significance of inputs or outputs, naturally, physical reservoir computing is much more concerned with physical meanings. This is particularly true in the context of robotics as we hope to gain insight into how to design better robot bodies to be exploited as computational resources.

For example, the input is not just an abstract time series, but is a real physical quantity that changes over time. The most natural interpretation in the light of the proposed models, i.e. using mass–spring–damper systems, inputs are forces that work on the body of the robot. This could be through the interaction of the robot with the environment, e.g. by grasping an object or by locomoting on the ground. But this could be also in the context of a sensor setup, where the sensor has to physically interact with the environment to gain information for measurements. Similarly, feedback signals (as in the feedback setup) can be forces obtained through the environment or through internal actuators that can change the state of the system.

Both, input and feedback, don't have to be restricted to forces but can be any physical entity that changes the state of the reservoir sufficiently to be picked up by the readouts. This also means that we don't have to use necessarily mechanical structures for the reservoir, but we could think also of chemical systems, electromagnetic interactions, etc. Of course, a combination of them is valid as well. Smart material and additive manufacturing, which both have gained a lot of traction recently,

will play a crucial role here. We refer the reader to Sects. 6 and 7 for a more in-depth discussion.

As previously pointed out in the introduction, one of the key properties of a reservoir is its exponential stability or in other terms fading memory. While in abstract reservoir computing setups one has to make sure to have stable systems (e.g. it's quite common to re-scale the connectivity matrix in ESNs), stability is a natural property in mechanical structures used for robotics, i.e. stability comes for "free."

Furthermore, there is a clear connection between material properties and the fading memory, i.e. how fast the system forgets its initial state. If we consider linear mass–spring–damper systems, the real part of the eigenvalue of the system matrix describes how fast, e.g. an impulse input is "faded out." More precisely, the exponential hull curve for the response of the system is defined by $e^{-d/m}$ with d being the linear damping factor and m the mass. This can be considered directly in the design process by choosing the right material or even by including mechanisms, e.g. through smart materials, to change the damping to adapt the reservoir and its corresponding fading memory for different tasks.

Another interesting aspect of physical reservoirs is that often there is not a clear border that separates them from the environment. Let's take the example of a fish-inspired underwater robot that we want to control for locomotion. The input (force) will come from some form of actuation in the system. This will change the dynamic state of the fish body. In addition, however, this also introduces changes in the water environment, which can reflect back onto the fish. This means the water is actually part of the reservoir and so are nonlinearities in the physical embodiment of the input (e.g. nonlinear features in the electrical motor) and the readout (e.g. nonlinear properties of the sensors). This points to the idea of embodiment, which states that a close interaction of body and environment is fundamental for the rise of intelligent behaviour (see, e.g. Pfeifer and Bongard 2006).

Another point of difference is that abstract reservoir computing approaches assume that the readout has access to the full state of the reservoir. However, practically, in physical reservoir computing setups in robotics most of the time this is not possible. It turns out that knowing the full state of the reservoir is not necessary and one can achieve working physical reservoir computing setups with a subset of the states.

In addition, it is quite typical in robotics to have a wider range of sensors measuring different quantities instead of measuring directly the state of the system. States are an abstract concept anyway and it's well established in linear control theory that systems can be easily transformed into others with the right matrix transformation. Practically, we can only measure real physical quantities, and the generic reservoir computing does not allow us any additional signal transformation at the readout. Practically, we have measurements that might provide redundant information. For example, two different gyroscopes at different locations on the robot will very likely produce partially redundant information. But also completely different sensors will potentially share information about states. Ideally, we want to reduce the amount of redundant information. The worst-case scenario would be linearly dependent readout

signals.[13] However, some overlap is fine since linear regression by definition will pick and choose (by assigning the right weights) the best signals to get as close as possible to the target.[14]

Another important difference is that physical reservoir computing approaches are naturally leading us to think in dynamical systems terms instead of abstract mathematical frameworks. This provides us with some intuition on how to build intelligent machines based on this concept. For example, Fuechslin et al. (2011) suggested to look at computation in the physical reservoir computing context as mechanical structures that implement attractor landscapes. This could help us to design physical structures with a bias, for example, by having implemented a number of potential attractor points and limit cycles (which would be different behaviours—e.g. various locomotion gaits) and corresponding readouts/feedbacks can exploit them. For a discussion of the potential of this approach, we refer to Hauser and Corucci (2017).

3 Example Cases from Robotics

There are a number of examples from the literature that have demonstrated the usefulness of applying physical reservoir computing in robotics. The implemented computations include abstract mapping (as a proof-of-concept) to applications in sensing and controlling.

We will first present results from simulations and then discuss real-world platforms.

In the original work by Hauser et al. (2011), a number of example computations have been presented that might be interesting in the context of robotics. The results range from learning to emulate a simple Volterra series,[15] a model of a nonlinear pendulum, inverse dynamics of a two-link robot arm, and NARMA systems taken from Atiya and Parlos (2000) that have been known to be hard to emulate with recurrent neural networks due to their long-term dependencies, see for example Hochreiter and Schmidhuber (1997).

An extension of the robot arm example has been introduced in Hauser and Griesbacher (2011), where the mechanical reservoir is directly connected to the two-link robot arm. The setup was able to learn to control itself to move along a desired (figure eight) end-point trajectory. Later, similar results were achieved using L-systems to "grow" the structure around the arm (Bernhardsgrütter et al. 2014).

---

[13] Note that this is directly connected to the degradation of the signal separation property discussed in Sect. 2.1, see Hauser et al. (2014) for a discussion.

[14] Note that linear regression is solving the optimization problem to find a set of optimal weights $\mathbf{w}^*$ that minimize the quadratic error between the target output $y_t(t)$ the produced output $y(t)$.

[15] The reason being that the theoretical model from Hauser et al. (2011) is based on Volterra series and it's a generic way to approximated nonlinear, exponentially stable sets of differential equations.

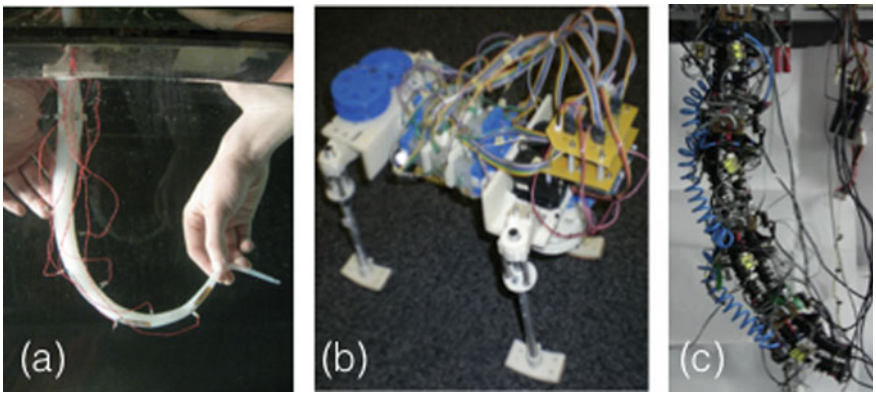

Fig. 2 Example cases with real physical platforms. **a** Octopus arm setup used in Nakajima et al. (2018a) and (Nakajima et al., 2013), **b** Kitty robot from Pfeifer et al. (2013), **c** Pneumatically driven, modular robot arm from Eder et al. (2017)

As pointed out before in Sect. 2, Hauser et al. (2012) also introduced a second theoretical model dealing with physical reservoir approaches that consider linear feedback loops from the output back into the reservoir. In the same work, a number of example cases were presented. They ranged from learning to emulate robustly various nonlinear limits cycles (e.g. like the van der Pol equations and others) to switching between different gaits by changing the readout weights. Finally, they also learned to produce three different limit cycles with the same set of readout weights. The change from one to another was only driven by a change in input. Specifically, the otherwise constant input force was switched to one of three different constant values. This means, depending on how strong the physical reservoir was squeezed, it reacted with a different limit cycle. This example is particularly interesting as it points to the possibility to change behaviour (e.g. gaits) triggered by changes in the environment (through the change in forces acting on the robot, e.g. by putting a heavy weight on it). For a deeper discussion of this results, we refer to Hauser et al. (2014) and Fig. 2.

Caluwaerts et al. (2013) were able to show that the complex dynamics of worm-like structures built based on the tensegrity principles can be used to produce robustly control signals for the locomotion of the robot. They used also learning to optimize the locomotion behaviour. In addition, the work showed that the body can be used as a sensor to classify the ground (flat vs bumpy) by using it as a physical reservoir. Another simulation work showing the capability of classification with the help of mechanical structures was introduced by Johnson et al. (2014). They used the same mass–spring–damper systems as suggested by Hauser et al. (2011) to build a system that can actively discriminate different shapes.

Still in simulation, but with a different approach to physical reservoir computation in robotics has been suggested by various people by using more structured reservoirs. Instead of random networks, bio-inspired morphologies were used. Naturally, they

are more constrained with respect to their potential computational power, but they work surprisingly well. For example, Sumioka et al. (2011) used a mechanical setup that was loosely inspired by the muscle–tendon–bone arrangements in the biological system as a reservoir. Despite the simplicity of the model, they were able to emulate a Volterra series. Nakajima et al. (2013) built a network of nonlinear mass–spring–damper systems simulating an octopus arm. The input was applied at the shoulder and the readout was obtained by measuring the strain throughout the body, i.e. similar to proprio-receptive sensing.

More recently, even more structured setups have been shown to work as a reservoir as well. Yamanaka et al. (2018) demonstrated that a 2D grid structure, representing a model of a soft cloth, can be used as a reservoir as well. In this particular work, the authors investigated which stiffness and damping values lead to better performances for various computational tasks.

While simulation results are interesting in themselves, robotics is a field of research that ultimately wants to develop frameworks and technologies that can be used under real-world conditions. The same is true for physical reservoir computing in robotics. The first step[16] has been made by a series of work by Nakajima et al. They used a silicone-based octopus-inspired robot arm as a reservoir. The input was in form of rotation of the arm introduced at the shoulder via an electric motor (compare Figs. 2a and 3a). The readout was obtained via strain sensors along both sides of the octopus arm. Nakajima et al. were able to demonstrate that this setup was able to learn to emulate timers and various digital functions like delays and parity (demonstrating the memory capacity of the system), see Nakajima et al. (2014). In the same work, it has been even shown that the setup is able to learn to produce a control signal for itself, i.e. the body of the octopus arm is used as a computational resource (i.e. reservoir) to control robustly the movement of the same arm. Interestingly, when they interacted with the arm (touching and grasping it during the movement), the arm started to react to it by naturally looking movements, i.e. it looked as if it tried to wriggle itself free and sometimes it stopped completely until it was released. While this interpretation is clearly anthropomorphic, it nevertheless points to the fact that the feedback loop through the environment plays a crucial role in the behaviour in such a setup.

Nakajima et al. also emulated with the same setup various NARMA systems with different complexities and other nonlinear, dynamical mappings (Nakajima et al. 2018a, b) with the goal to systematically explore the limitations of the computational power, e.g. they quantified the amount of available fading memory by comparing it to standard ESN.

Zhao et al. (2013) applied the idea of physical reservoir computing to locomotion. The physical reservoir was constructed out of a bio-inspired assemble of hard and soft parts (resembling loosely the spinal structure) and randomly distributed pressure sensors (compare Figs. 2b and 3b). The system was able to robustly locomote

[16] Note that there had been previous work on physical reservoir computing, e.g. the previously mentioned "bucket in the water" setup by Fernando and Sojakka (2003), but none in the context of robotics.

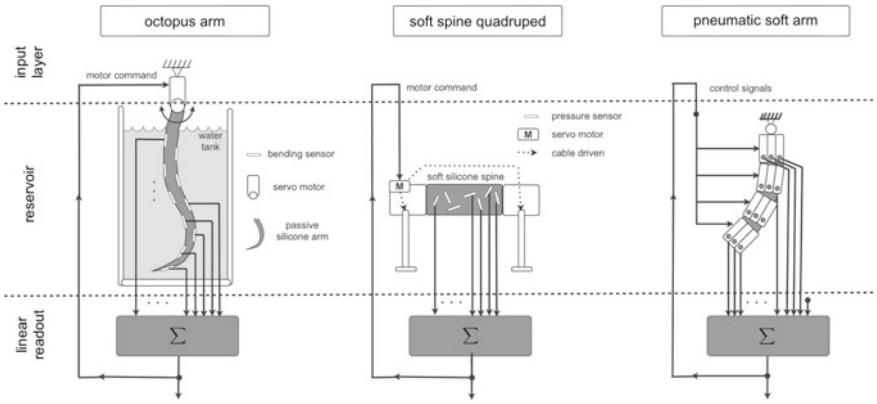

Fig. 3 Schematics setups for real platform shown in Fig. 2 and discussed in the text. Also compare to Fig. 1. Figures adapted from Hauser et al. (2014) and Eder et al. (2017)

and change direction. Another locomotion example on a real-world platform was provided by Caluwaerts et al. (2014) using a mechanical tensegrity structure to learn an oscillator to control it.

Physical reservoir computing has also been shown to work for other tasks as well. For example, Eder et al. (2017) used a pneumatically driven robot arm and its nonlinear dynamics as a reservoir to control the arm (compare Figs. 2c and 3c). While the results were interesting, they also revealed clearly limitations for using this approach on this particular platform.

Besides the mapping of abstract computations and to learn to control a system, real physical reservoir computing setups have also been used now for sensing, classification, and even modelling. For example, Soter et al. (2018) used a physical reservoir computing setup extended by an RNN to predict the movement (actually the pixels of the video of the movement) of an octopus arm solely based on the proprio-receptive information (bending). It learns "imaging" the visual movement of the arm through observing its internal states.

Recent work by Judd et al. (2019) was able to use physical reservoir computing to detect objects in the vicinity of the moving octopus arm without touching them. This included classification, i.e. is the object there or not, as well regression, i.e. estimation of where it is located. The results point to the fact that the environment indeed is part of the reservoir and should play a crucial role when developing intelligent systems (sensing and control) with physical reservoir computing setups in robotics.

In summary, there is a substantial amount of exciting results, however, there is also still a wide range of interesting, yet to be explored ideas. To inspired future researcher working in the field, we discuss in the next section what advantages a physical reservoir computing approach in robotics might have.

4 Advantages of Physical Reservoir Computing in Robotics

The application of physical reservoir computing in robotics has a number of remarkable aspects and advantages.

For example, the readout for conventional reservoir computing is typically the same for every state, since it's an abstract mapping. However, in a physical setup, we need real physical sensors to obtain the information about the state of the system. As previously pointed out, it's not trivial to get the information about the full state in a real robot. However, it turns out that is also not needed. Since the readout weights are found with the help of linear regression, we can provide a variety of sensory signals. Linear regression automatically chooses the "best" ones, i.e. the ones that provide the best information to reduce the error between output and target signal. This also means that linear regression doesn't "care" what kind of signal we provide or where it comes from. This means we can combine sensory information from gyroscopes, pressure sensors, stretch sensors, and even binary switches. We can even include and combine sensors that measure in completely different physical domains, e.g. a force sensor with sensors to measure chemical concentration or temperature. If they provide useful information, they will get high enough readout weights assigned.[17]

Another advantage is the inherent robustness of the approach. Examples in Hauser et al. (2012) demonstrated that the learned limit cycles, although only provided with data from a certain range of the state space, seem to exhibit global stability. We believe this is due to the inherent global stability of the underlying mass–spring–damper systems. Obviously, practically there are limitations with respect to global stability, after all, mechanical structures will eventually break under too high forces. Nevertheless, this points to the underlying property of inherent stability, which can be exploited. Maybe frameworks like contraction theory (Lohmiller and Slotine 1998) or concepts like passivity from control theory, especially more recent extensions like Forni and Sepulchre (2019), might be able to provide us with a general proof of this intuition.

Another interesting aspect of robustness is that smaller changes in the reservoir, like tearing of parts of the robot because of rough use or interaction with the environment, can be easily counteracted by slightly changing the weights of the readout layer. As a result, imprecise design is not a problem. We don't need a perfectly controlled fabrication process as long as the final result is close enough for being useful. This points to a very different way to build robots and it suggests a quite radical break with conventional robotic design approaches (see discussion later in Sect. 7).

Another advantage of reservoir computing in general is that learning through applying linear regression is very fast. This is particularly important for robotics. Also, online learning algorithms are thinkable using the wide range of available tools that implement recursive, online versions of linear regression.

Another particularly interesting advantage is that noise is beneficial for the reservoir computing setup that uses feedback. It's common practice in reservoir computing in general to add noise to learning data when using feedback loops. This helps to

[17] Note that a limitation is that the different sensors should provide linearly independent information.

learn instead of a simple trajectory (e.g. of a limit cycle) an attractor region around this trajectory making it more robust. Interestingly, while in simulations, noise has to be added artificially, it seems the noise present in real physical robots is enough and appropriate.

Finally, looking at robot design from the viewpoint of reservoir computing leads to novel approaches to build robots and might be a way around the implicit assumption in the robotics community that we have to build robots such that we can easily model them. In the context of physical reservoir computing underactuation, i.e. degrees of freedom or states of the system that are not directly controlled by the input are a prerequisite for a computationally powerful body. Hence, using more compliant and soft structures for building robots might be beneficial. A more detailed discussion of this point is carried out in Sect. 6.

Besides the listed advantages, clearly, the physical implementation of reservoirs in robot bodies also implies a range of problems and limitations, which are discussed in the next section.

5 Limitations of Physical Reservoir Computing in Robotics

In general, the implementation of a reservoir in a real, physical body also always means the introduction of physical limitations. While abstract approaches, e.g. like ESNs, in principle, don't have to worry about limitations (assuming we have the right dynamical properties to make a useful reservoir), a real body also means real-world constraints. For example, (stable) mechanical structures always have the effect of a low-pass filter. Consider a soft structure, i.e. a body with low stiffness. A high-frequency input will not get transmitted effectively enough through the body. This means whatever information is provided by this particular input is lost and the reservoir is not useful for this kind of computation. On the other hand, very stiff structures have no problem to transmit high-frequency vibration throughout their bodies. But this also means it has a very low fading memory as every effect is almost immediately damped out.

Another limitation, which is inherent to any reservoir computing setup in general, is that we don't directly control what is happening in the reservoir. The readout uses linear regression to pick and choose the best signals; however, we have very little control over how to improve the performance of the reservoir. There has been some work on improving the performance through changing the structure, but so far almost none of them have been implemented in real physical systems. We discuss the potential of such approaches later in Sects. 6 and 7.

In general, the lack of control over the dynamics of the reservoir is a big point of discussion in a robotics application. Conventional approaches to design and control robots are particularly keen to build systems that are easily modelled and, therefore, easily controlled. However, this reduces the computational power of the corresponding morphology and therefore makes them unfit to serve as reservoirs. This is an interesting tension which is particular to robotics and it's not clear if these two seem-

ingly opposite positions can be (or even should be) reconciled. However, since both have advantages and disadvantages, most likely both will be used as complementary approaches and they will be implementing different tasks and at different levels in future robotic systems.

Connected to that is the problem of safety. If we don't have a good enough model of the dynamics of the reservoir, we can't prove the safety or stability of the learned behaviour. Again, biological systems don't seem to have a problem with that and maybe the previously mentioned inherent (seemingly global) stability (see Sect. 4) might be the key to this problem.

Finally, reservoir computing in principle is based on supervised learning. This means we have to provide the right input–output data set. While this might be possible for sensor applications, this is quite tricky for feedback-based setups, e.g. in locomotion. We often don't know the best mapping that we should learn to emulate. One possible solution is to start with a good enough guess and use online adaptation mechanisms to improve on them, see, e.g. Caluwaerts et al. (2014).

6 Connection to Soft Robotics

As pointed out before, one of the main motivations to use nonlinear mass–spring–damper systems as a basic building block for the models of mechanical reservoirs was the observation that soft-bodied robots, as well as bodies of biological systems, can be described elegantly by a network of such systems.

However, there is a much stronger connection between physical reservoir computing and soft robotics if we have a closer look at the dynamical properties of soft-bodied structures.[18] They typically exhibit complex, nonlinear dynamics, a high-dimensional state space, underactuation, and noise. These properties are all perceived as negative in conventional robotics since they make it more difficult to model and consequently control such systems. However, on the other side, these are the exact properties that are needed to build computationally powerful reservoirs. So, instead of avoiding complexity in the dynamics of robots, through the framework of reservoir computing, we can embrace them and, consequently, exploit them. One could even claim that reservoir computing is a good candidate to solve the control problem in soft robotics, see, e.g. Hauser (2016) for a discussion.

In addition, the rise of soft robotics is strongly coupled with the advancements in material science and additive manufacturing. These two fields of research have been proven over and over again to be able to extend the already rich set of dynamically interesting building blocks. Most of them have the potential to eventually be exploited in the context of physical reservoir computing. Most of the so-called smart materials have highly interesting, nonlinear dynamic behaviours that are potentially

[18] In the context of Sect. 6, we discuss only soft robotics structures. However, all the points are also true for biological systems or hybrid structures, e.g. the mixture of soft artificial structures and biological tissue.

beneficial to boost the computational power of a reservoir. In addition, the input can be "perceived" by the reservoir through mechanisms offered by smart materials. Direct physical interaction with the environment can be translated into changes in the dynamics. For example, the stiffness can change in some parts of the body through physical interaction or a soft robot "bumping" into an object can serve as a reliable input for the setup to switch behaviour, i.e. to switch to a different attractor space.

Another important point of connection between soft robotics and physical reservoir computing is that both research fields emphasize the importance of the embodiment. While conventional robotics tries to reduce the influence of the body of the robots (and the environment) as much as possible, soft robotics embraces the idea to outsource functionality to the body—and so does physical reservoir computing.

Finally, smart materials have the great potential to serve as substrates to implement future technologies that are capable of extending the notion of physical reservoir computing. This includes optimization of the reservoir and learning in material. Some of these future directions are discussed in the following chapter.

7 The Future of Physical Reservoir Computing in Robotics

While there has been quite a lot of excitement around the idea of outsourcing computation to the body of robots in the form of a reservoir, there is still a large underexplored potential pointing to a number of interesting research questions and corresponding applications.

Foremost, reservoir computing, physical or not, naturally raises the question about how to optimize the reservoir. While it's a clear advantage that we don't have to adapt parameters in the reservoir during learning, it also limits our control. Linear regression always finds the optimal solution based on the signal (readouts) it receives. However, we don't know how to get better or more information out of our reservoir. Furthermore, as previously pointed out in Sect. 2, the existing theoretical frameworks don't give us any specific design guidelines. We only get very high level suggestions of what kind of properties we should be looking for in a reservoir—i.e. high-dimensionality, underactuation, and nonlinear dynamics.[19] For a given computational task, e.g. a specific nonlinear controller or specific sensory filter that we want to emulate, the models don't give us a direct mapping to (in some sense) optimal reservoir. We don't know how many mass–spring–damper systems (or systems of equivalent complexity) are needed to achieve a certain performance. Therefore, typically, reservoirs are initialized randomly (in simulation) or pre-existing robotic structures, which had been designed with another functionality in mind, are simply exploited as a reservoir.

Since optimizing the reservoir is a general challenge that people are addressing, we might be able to draw inspiration from their results for specific applications in robotics. For example, in the context of LSMs, some approaches have been sug-

[19] And noise in the case of external feedback loops.

gested. Sussillo and Abbott proposed the First-Order Reduced and Controlled Error (FORCE) algorithm (Sussillo and Abbott 2009) where they adapt synaptic strengths by feeding back the error. Another approach in the context of LSMs, in this case, based on reward-modulated Hebbian learning, has been suggested by Hoerzer et al. (2014). They use a stochastic approach for the optimization of the reservoir. In the case of LSMs, which are inspired by brain structures, there is a clear connection to biological adaptation mechanisms, i.e. neuro-plasticity. However, in the case of physical reservoir computing in robotics, the picture becomes more blurry as there are many different ways to change real physical, morphological features, e.g. ranging from simply changing specific parameters like stiffness to complex growing processes. To the best of our knowledge, so far, there have been only simulation results to address this issue. For example, Hermans et al. (2014, 2015) proposed to use backpropagation through mechanical structures leading to optimal physical reservoirs. While their results are impressive, it would be great to see their approaches implemented in real physical bodies. While this has been out of the reach for a long time, mostly due to practical limitations, with the recent rise of soft robotics and additive manufacturing, we suddenly have a much broader set of tools available to build physically adaptive systems. There is a big potential for a range of interesting research directions with respect to learning directly in materials to improve the reservoir. Also, research in protocells, synthetic biology, and even biology (e.g. stem cells) is all very likely to be able to contribute to this idea. In general, this points clearly to much more sophisticated physical bodies and we might have to let go of the idea that we need to control explicitly every single aspect of a robotic system.

Another aspect connected to adaption in the reservoir is that we need more performance measurements that can help us to guide the adaptation process. For a lot of interesting tasks in robotics, we don't have a clear-cut target function. For example, if we have a given body and we want to exploit it for locomotion, we don't know how the optimal input–output mapping should look like. However, meaningful performance measurements can guide a corresponding adaptation process, either directly in the body/reservoir and/or at the readout. One promising approach is to use abstract measurements based on information-theoretic approaches. For example, for the measurements based on the concept of Predictive Information, see, e.g. work by Martius et al. (2013) and Martius (2014). Another possibility, suggested by Ghazi-Zahedi and Rauh (2015) and Ghazi-Zahedi et al. (2017), could be to measure directly how much computation is outsourced to the morphology (in our case the physical reservoir). These are just a few examples, but of course, there is a very strong body of work on statistical approaches in general. A reinterpretation in the light of physical reservoir computing might provide a number of exciting new approaches.

The idea of changing the body to improve the reservoir has an even wider implication. If we consider morphological structures (e.g. in form of mechanical systems) as being capable of representing computational functionality, then concepts like self-assembly, growing, and self-healing become an entirely new, additional meaning: Changing the body is changing the computational functionality. Instead of simply assembling to a morphological structure, which is interesting in itself, we also build an underlying computational functionality. We assemble also a program. The same

is also true for growing (Hauser 2019). A system (biological or artificial) capable of growing is able to construct functionality through this process. In addition, if such a system has self-healing capability, it wouldn't simply fix a broken leg, but also the underlying computational functionality.

In this context, it is interesting to note that there are many different ways to *build* a reservoir. One can easily image two reservoirs that are morphologically different but have the same computational power. This implies the growing process might have a primary goal, for example, to make a limb for locomotion, while a secondary goal (with less constraints) could be to construct a reservoir structure. As shown in real-world physical reservoir computing setups with existing robotic prototypes, already existing physical bodies can be exploited as reservoir even if during their design and building process this has not been considered at all.

Another point of discussion is how, in robotic applications, can we get away from the more homogeneous structures present in classical reservoir computing approaches. Traditional reservoir computing uses one simple module (e.g. an integrate and fire models in ESN, or spiking neural models in LSM) and then connects them with each other. However, the fundamental building blocks are always the same. In the context of physical reservoir computing and robotics, this seems to be quite an unnatural approach. Intelligent systems are made of a variety and dynamically different body parts. This again can be of great benefit if exploited properly. Unfortunately, so far this has not been really explored in the community.

One way to investigate this idea could be to include buckling mechanisms or hysteresis—both are quite common and naturally occurring dynamical features. Buckling could potentially facilitate the learning of different attractors (e.g. different gaits as in Hauser et al. 2012 or switching between different controllers as in Füchslin et al. 2013). There has also been very interesting work by Kachman et al. (2017) who showed from the viewpoint of thermodynamics that dynamical systems with two (or more) stable states can lead to self-organization under (thermal) noise. Hysteresis could also be useful, for example, to help to detect changes in the direction of the input. Note that real muscles have hysteresis and even more complex memory effects, see, for example Paetsch et al. (2012).

Another field of application of physical reservoir computing in robotics has been so far under-explored, i.e. the implementation of dynamically complex sensors. While there has been some work, see Sect. 3, there are still a lot of interesting open research questions and potential applications. One can think of the implementation of interesting (signal processing) filters, information-based approaches, or adaptive sensing morphologies that change the sensing modality based on different tasks. This is especially interesting if we consider adaptation mechanisms to optimize the physical reservoir (i.e. morphology of the sensor) to improve the sensing performance. Information-theoretic approaches and research on sensing systems in biological systems (especially insects) will play a crucial role in this context. Also, the idea of building more structured morphologies for the reservoir is a valid angle to be explored. For example, the project "Computing with Spiders' Webs" (Hauser and Vollrath 2017) works on spider web-inspired approaches to build physical reservoir computing sensors to measure vibration and flow.

As one can see there is still a great, unexplored potential in the approach of physical reservoir computing in the context of robotics. The field is rich with research opportunities and we are looking very much forward to see novel approaches, more robots, and exciting research results from the community.

Acknowledgements This publication has been written with the support of the Leverhulme Trust Research Project RPG-2016-345.

References

A.F. Atiya, A.G. Parlos, New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE Trans. Neural Netw. **11**(3), 697–709 (2000)

R. Bernhardsgrütter, C.W. Senn, R.M. Füchslin, C. Jaeger, K. Nakajima, H. Hauser, Employing L-systems to generate mass-spring networks for morphological computing, in *Proceedings of International Symposium on Nonlinear Theory and its Applications (NOLTA2014)*. Research Society of Nonlinear Theory and its Applications, IEICE (2014), pp. 184–187

S. Boyd, Volterra series: engineering fundamentals. PhD thesis, UC Berkeley (1985)

S. Boyd, L. Chua, Fading memory and the problem of approximating nonlinear operators with Volterra series. IEEE Trans. Circuits Syst. **32**(11), 1150–1161 (1985)

K. Caluwaerts, M. D'Haene, D. Verstraeten, B. Schrauwen, Locomotion without a brain: physical reservoir computing in tensegrity structures. Artif. Life **19**, 35–66 (2013)

K. Caluwaerts, J. Despraz, A. Işçen, A.P. Sabelhaus, J. Bruce, B. Schrauwen, V. SunSpiral, Design and control of compliant tensegrity robots through simulation and hardware validation. J. R. Soc. Interface **11**(98), 20140520 (2014)

M. Eder, F. Hisch, H. Hauser, Morphological computation-based control of a modular, pneumatically driven, soft robotic arm. Adv. Robot. **32**(7), 375–385 (2017)

C. Fernando, S. Sojakka, Pattern recognition in a bucket, in *Advances in Artificial Life SE - 63*, vol. 2801, Lecture Notes in Computer Science, ed. by W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, J.T. Kim (Springer, Berlin, 2003), pp. 588–597

F. Forni, R. Sepulchre, Differential dissipativity theory for dominance analysis. IEEE Trans. Autom. Control. **64**(6), 2340–2351 (2019)

R.M. Füchslin, H. Hauser, R.H. Luchsinger, B. Reller, S. Scheidegger, Morphological computation: applications on different scales exploiting classical and statistical mechanics, in *Proceedings of the 2nd International Conference on Morphological Computation*, September 2011, ed. by R. Pfeifer, S. Hidenobu, R.M. Füchslin, H. Hauser, K. Nakajima, S. Miyashita (2011)

R.M. Füchslin, A. Dzyakanchuk, D. Flumini, H. Hauser, Morphological computation and morphological control: Steps toward a formal theory and applications. Artif. Life **19**, 9–34 (2013)

K. Fujii, K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning. Phys. Rev. Appl. **8** (2017)

K. Ghazi-Zahedi, J. Rauh, Quantifying morphological computation based on an information decomposition of the sensorimotor loop, in *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, vol. 27 (2015), pp. 70–77

K. Ghazi-Zahedi, C. Langer, N. Ay, Morphological computation, synergy of body and brain. Entropy **19**(9) (2017)

H. Hauser, Morphological computation – a potential solution for the control problem in soft robotics, in *Proceedings of the 19th International Conference on CLAWAR 2016* (2016)

H. Hauser, Resilient machines through adaptive morphology. Nat. Mach. Intell. **1**(8), 338–339 (2019)

H. Hauser, G. Griesbacher, Moving a robot arm by exploiting its complex compliant morphology, in *Proceedings of the 2nd International Conference on Morphological Computation*, September 2011, ed. by R. Pfeifer, S. Hidenobu, R.M. Füchslin, H. Hauser, K. Nakajima, S. Miyashita (2011)
H. Hauser, F. Corucci, Morphosis - taking morphological computation to the next level. Biosyst. Biorobot. **17**, 117–122 (2017)
H. Hauser, F. Vollrath, Leverhulme Trust Research Project RPG-2016-345, Computing with Spiders' Webs (2017)
H. Hauser, A.J. Ijspeert, R.M. Füchslin, R. Pfeifer, W. Maass, Towards a theoretical foundation for morphological computation with compliant bodies. Biol. Cybern. **105**, 355–370 (2011)
H. Hauser, A.J. Ijspeert, R.M. Füchslin, R. Pfeifer, W. Maass, The role of feedback in morphological computation with compliant bodies. Biol. Cybern. **106**(10), 595–613 (2012)
H. Hauser, K. Nakajima, R.M. Füchslin, Morphological computation – the body as a computational resource, in *E-book on Opinions and Outlooks on Morphological Computation*, ed. by H. Hauser, R.M. Füchslin, R. Pfeifer (2014), pp. 226–244
M. Hermans, B. Schrauwen, P. Bienstman, J. Dambre, Automated design of complex dynamic systems. PLoS ONE **9**(1) (2014)
M. Hermans, M. Burm, T. Van Vaerenbergh, J. Dambre, P. Bienstman, Trainable hardware for dynamical computing using error backpropagation through physical media. Nat. Commun. (2015)
S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
G.M. Hoerzer, R. Legenstein, W. Maass, Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. Cereb. Cortex **24**(3), 677–690 (2014)
K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators. Neural Netw. **2**, 359–366 (1989)
A. Isidori, *Nonlinear Control Systems*, third edn. (Springer GmbH, 2001)
H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
C. Johnson, A. Philippides, P. Husbands, Active shape discrimination with physical reservoir computers, in *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, vol. 14 (2014), pp. 176–183
T. Kachman, J.A. Owen, J.L. England, Self-organized resonance during search of a diverse chemical space. Phys. Rev. Lett. **119** (2017)
W. Lohmiller, J.-J.E. Slotine, On contraction analysis for non-linear systems. Automatica **34**(6), 683–696 (1998)
W. Maass, T. Natschlaeger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)
W. Maass, P. Joshi, E.D. Sontag, Computational aspects of feedback in neural circuits. PLoS Comput. Biol. **3**(1), e165 (2007)
G. Martius, R. Der, N. Ay, Information driven self-organization of complex robotic behaviors. PLOS ONE **8**(5), 1–14 (2013)
G. Martius, L. Jahn, H. Hauser, V. Hafner, Self-exploration of the stumpy robot with predictive information maximization, in *From Animals to Animats 13*, ed. by A.P. del Pobil, E. Chinellato, E. Martinez-Martin, J. Hallam, E. Cervera, A. Morales. Lecture Notes in Computer Science, vol. 8575 (Springer International Publishing, 2014), pp. 32–42
K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D.G. Caldwell, R. Pfeifer, A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. Front. Comput. Neurosci. **7**(91), 91 (2013). Research Topic: Modularity in Motor Control: From Muscle Synergies to Cognitive Action Representation
K. Nakajima, T. Li, H. Hauser, R. Pfeifer, Exploiting short-term memory in soft body dynamics as a computational resource. J. R. Soc. Interface **11**(100), 20140437 (2014)
K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Exploiting the dynamics of soft materials for machine learning. Soft Robot. **5**(3) (2018a)

K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Exploiting the dynamics of soft materials for machine learning. Soft Robot. (2018b)

C. Paetsch, B.A. Trimmer, A. Dorfmann, A constitutive model for activepassive transition of muscle fibers. Int. J. Non-Linear Mech. **47**(2), 377–387 (2012)

R. Pfeifer, J.C. Bongard, *How the Body Shapes the Way We Think* (The MIT Press, 2006)

R. Pfeifer Q. Zhao, K. Nakajima, H. Sumioka, H. Hauser, Spine dynamics as a computational resource in spine-driven quadruped locomotion, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2013)

B. Schrauwen, D. Verstraeten, J. Van Campenhout, An overview of reservoir computing: theory, applications and implementations, in *Proceedings of the 15th European Symposium on Artificial Neural Networks* (2007), pp. 471–482

Y. Shim, P. Husbands, Feathered flyer: integrating morphological computation and sensory reflexes into a physically simulated flapping-wing robot for robust flight manoeuvre, in *ECAL*, ed. by F. Almeida e Costa et al. (Springer, Berlin/Heidelberg, 2007), pp. 756–765

J.J. Slotine, W. Lohmiller, Modularity, evolution, and the binding problem: a view from stability theory. Neural Netw. **14**(2), 137–145 (2001)

A. Smerieri, F. Duport, Y. Paquot, B. Schrauwen, M. Haelterman, S. Massar, Analog readout for optical reservoir computers, in *Advances in Neural Information Processing Systems*, vol. 25, ed. by F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Curran Associates, Inc., 2012), pp. 944–952

G. Soter, A. Conn, H. Hauser, J. Rossiter, Bodily aware soft robots: Integration of proprioceptive and exteroceptive sensors, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018 (2018), pp. 2448–2453

G. Soter, A. Conn, H. Hauser, J. Rossiter, Sensing through the body – non-contact object localisation using morphological computation, in *2018 IEEE International Conference on Soft Robotics (RoboSoft)* (2019)

H. Sumioka, H. Hauser, R. Pfeifer, Computation with mechanically coupled springs for compliant robots, in *IEEE International Conference on Intelligent Robots and Systems* (IEEE, 2011), pp. 4168–4173

D. Sussillo, L.F. Abbott, Generating coherent patterns of activity from chaotic neural networks. Neuron **63**(4), 544–57 (2009)

V.N. Vapnik, *Statistical Learning Theory* (Wiley, New York, 1998)

Y. Yamanaka, T. Yaguchi, K. Nakajima, H. Hauser, Mass-spring damper array as a mechanical medium for computation, in *Artificial Neural Networks and Machine Learning – ICANN 2018*, ed. by V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, I. Maglogiannis (Springer International Publishing, Cham, 2018), pp. 781–794