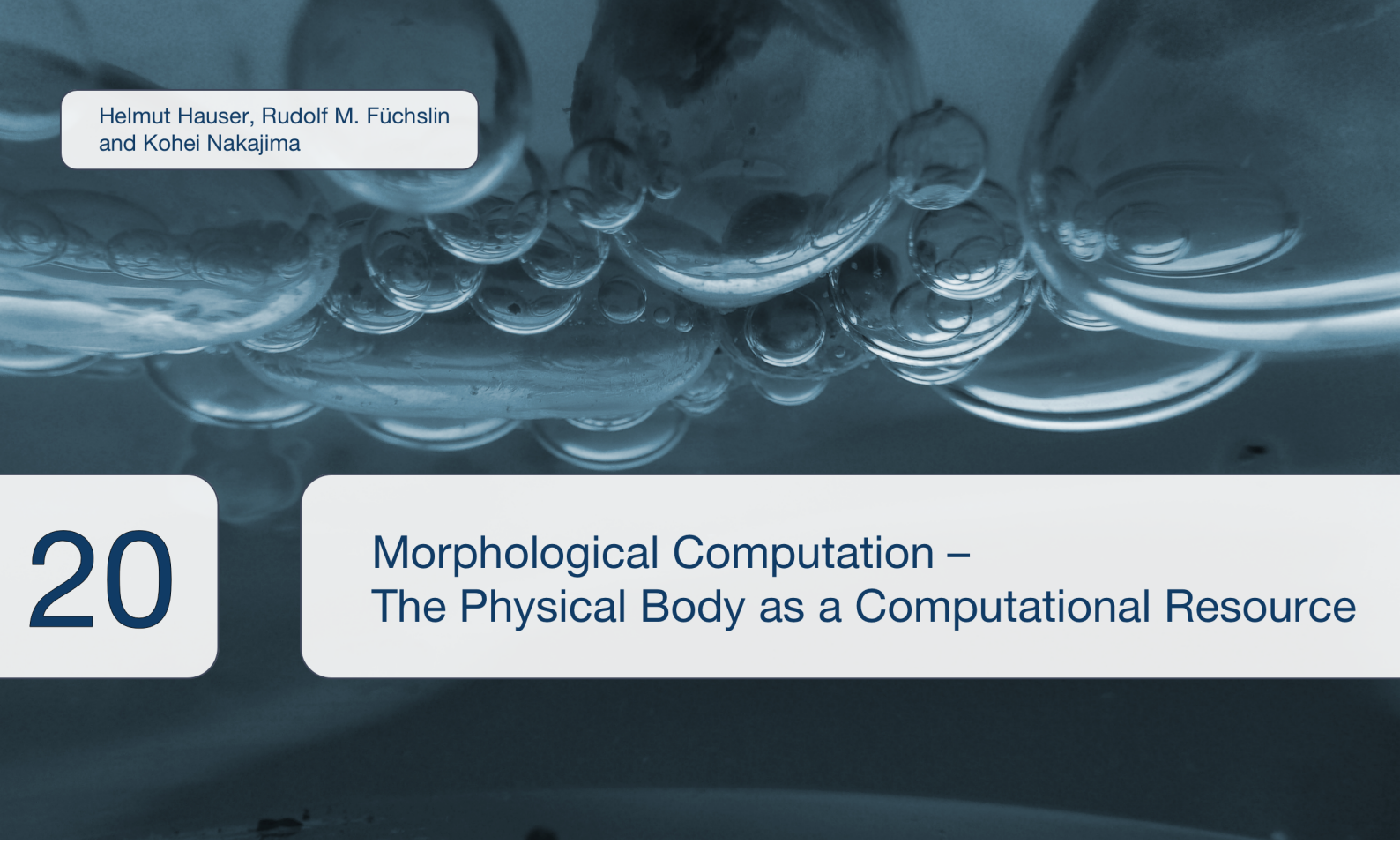
Helmut Hauser, Rudolf M. Füchslin
and Kohei Nakajima



20

Morphological Computation – The Physical Body as a Computational Resource

Abstract: *Recently, two theoretical models for morphological computation have been proposed [13,14]. Based on a rigorous mathematical framework and simulations it has been demonstrated that compliant, complex physical bodies can be effectively employed as computational resources. Even more recent work showed that these models are not only of theoretical nature, but are also applicable to a number of different soft robotic platforms. Motivated by these encouraging results we discuss a number of remarkable implications when real physical bodies are employed as computational resources.*

Introduction

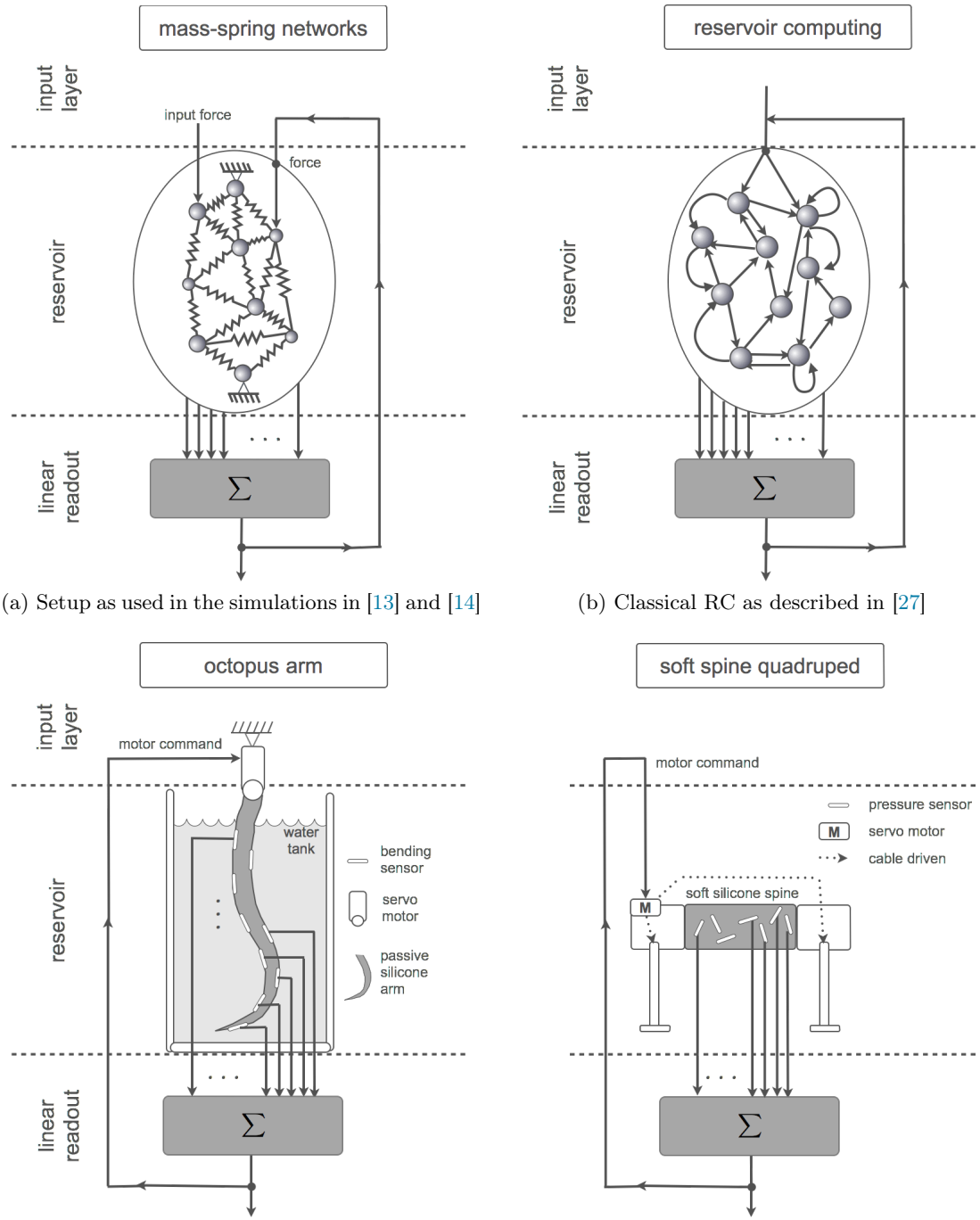
The two underlying theories of [13] and [14] are based on two very different theoretical bases. The first one [13] employs theoretical results by Boyd and Chua [4] from signal processing and discusses mathematical operators as a form to describe computation. Such operators are mathematical objects that can, for example, map (input) functions onto (output) functions.[1] In the context of robots we talk about functions in time, for example, the mapping from sensory input streams onto sensory output streams.

The second model introduced by Hauser et al. [14] is based on a tool from nonlinear control theory (i.e., feedback linearization, see [17] for more details) and, consequently, uses nonlinear dynamical systems to describe computation. Note that smooth non-autonomous dynamical systems, wherein the input function is interpreted as a non-autonomous driving term, are mathematical operators as well, since they map input functions in time onto output functions in time.

Interestingly, despite the obvious differences in their choices of mathematical tools for the two theories, the two derived corresponding computational setups look quite similar. Figure 1a shows the considered setup for the second theoretical model, while the setup based on the first theoretical model is very similar, it simply lacks the feedback loop. The reason is that both mathematical models are motivated by the same machine learning technique called reservoir computing (RC). This is a supervised learning approach, which has been quite successful in a number of challenging tasks including nonlinear mapping of input streams onto output streams (i.e., learning to emulate complex, nonlinear differential equations). A good overview of its success story can be found in [27].

At the core of RC lays the so-called *reservoir*, a randomly initialized high-dimensional, nonlinear dynamical system, which maps the typically low-dimensional input (stream) onto its high-dimensional state space in a nonlinear fashion. In that sense the reservoir takes the role of a kernel (in the machine learning sense, i.e., the nonlinear projection of a low dimensional input into a high-dimensional space, see, e.g., [13] for a discussion). In addition, the reservoir, being a dynamical system, has the inherent property to integrate input information over time, which is obviously beneficial for any computation that needs information on the history of its input values. It is important to note that the reservoir is not altered during the learning process. Although it is randomly initialized, its dynamic parameters are fixed afterwards. In order to learn to emulate a desired input output behavior (to be more precise, a desired mapping from input streams to output streams), one has to add a linear output layer, which simply calculates a linearly weighted sum of the signals of the high-dimensional state space of the reservoir. These output weights are the only parameters that are adapted during the learning process. Figure 1b shows the classical RC setup, where the nodes represent simple, but nonlinear differential equations. They are randomly connected with each other and, therefore, build a complex, nonlinear dynamic system, i.e., the reservoir.

[1] In more general terms an operator is defined as the mapping from a vector space or module onto another vector space or module. Since functions can be understood as elements of a function space, which under the usual operation of addition and scalar multiplication form a vector space, the language and concepts of operator theory are applicable for the study of mappings from functions to functions.



(a) Setup as used in the simulations in [13] and [14]

(b) Classical RC as described in [27]

(c) Schematics of the setup of the octopus experiments described in [24, 23, 22]

(d) Schematic setup of the quadruped robot with soft spine from [34]

Figure 1: *Overview of different RC setups referred to in the text. (a) is the classical RC setup with generic nodes. (b-d) are schematics of implementations of Physical RC where real physical bodies are employed as reservoirs and, therefore, as a computational resources.*

The remarkable conclusion is that we can learn to emulate complex, nonlinear computations (e.g., complex operators like nonlinear dynamical systems representing, e.g., a nonlinear controller) by simply finding some linear and static output weights. Hence, the task to learn to emulate a *nonlinear dynamical system* is, with the help of the reservoir, reduced to *simple linear regression*.

Another remarkable property of the RC approach is that the reservoir is not bound to be chosen from a specific class of dynamical systems. It only has to exhibit a number of rather generic properties to be useful as a reservoir: It should be high-dimensional, nonlinear and exhibit stable temporal integration.[2] As a result there exist different flavors of RC, e.g., echo state networks [18] or liquid state machines [20], employing different types of artificial reservoirs. For more details we refer to [27].

In the more recent years the RC approach has outgrown its pure machine learning domain. Looking at the desired properties for the reservoir (i.e., it should be a complex, nonlinear dynamical system with time integration capabilities) one can see that these properties cannot only be found in artificial dynamical systems that have been designed explicitly with this purpose in mind. Actually, a number of real physical systems exhibit these features as well and, hence, can potentially serve as reservoirs. One of the first examples along this line of thought was the "water in the bucket" experiment of [8], where the dynamic, nonlinear effects of a water surface have been successfully exploited as a reservoir to carry out vowel classification. However, recently a whole series of applications, especially, in nonlinear optics and soft robotics have extended this notion quite impressively, even so far, as there has been coined the term Physical RC to describe this research approach. In this work we concentrate on a discussion on Physical RC in the context of robotics. For more details on applications in nonlinear optics, we refer to [1] or [28, 7].

The theoretical setups proposed in [13] and [14] fall also into this category. They provide the theoretical basis for a wide range of real physical systems that can serve as reservoirs. Even the presented (rather abstract) networks of nonlinear springs and masses (and their simulations) are meant to provide a generic description of real physical bodies capable of serving as computational resources (compare Figure 1a). A similar line of thought has been applied by Caluwaerts et al. [5] by simulating a tensegrity structure and use it as a physical reservoir. There exist also a number of simulations employing more concrete platforms. For example, Nakajima et al. [23] implemented a sophisticated bio-inspired simulation of an octopus arm and demonstrated its computational power. Sumioka et al. [29] simulated a rather simple model of a human musculoskeletal system and employed it successfully as a reservoir. Hauser and Griesbacher [12] used simulations to show that a compliant physical body can be used to control a rigid robot arm. Bernhardsgrütter et al. [2] extended this work by letting the soft body structure grow based on rules encoded as L-systems.

Next to the simulation results, which are already remarkable, a series of work has

[2] In RC literature temporal integration is referred to as the *fading memory property*. In the context of dynamical systems this means we need to have an exponentially stable system that has one point of equilibrium or stays "close enough" to such a point - for a proof see [3].

demonstrated the applicability to real physical platforms. For example, a soft silicone based octopus arm has been used by Nakajima et al. to carry out computations and to implement a feedback controller [22]. Zhao et al. [34] constructed a quadruped platform that employs a bio-inspired soft spine as a reservoir to control the locomotion of the robot. Even tensegrity structures have been built and employed in such a Physical RC approach with the goal to use them for exo-planetary exploration [6].

While one might understand Physical RC as a simple extension of the original RC approach, we strongly believe that there are a range of remarkable implications, especially, when you consider implementations in the context of morphological computation on real-world robotic platforms like soft robots. Suddenly, abstract terms and notions from machine learning have real-world, physical meanings. We believe, as a consequence, this initiates a radical change of viewpoint, which implies a paradigm shift with respect to robotic design and even to computational theory.

In the following sections we are going to highlight and discuss some of these implications and we hope to inspire researchers to enter this young and exciting field of research.

The Body is not a Computer, but ...

The original RC approach is a pure machine learning technique. Its main purpose is to do computation, or to be more precise, to provide a learning setup to emulate a desired computation. However, if you work with a real physical body as a reservoir you suddenly have to deal with a completely different point of view and, as a result, abstract terms (like fading memory, kernel property, etc.) have real world implications and interpretations. Typically, the body or (body parts) of biological systems and of robots have (a set of) specific functions and are not explicitly built for computation. For example, the leg is used for locomotion or to kick a ball. The hand is meant to grasp an object or to play piano. The Physical RC approach, however, implies that on top of these functions we can exploit the dynamical properties of these body parts to carry out relevant computational tasks (e.g., to provide an appropriate continuous control signal based on the continuous sensory streams to stabilize the movement when kicking the ball). One could even argue that we get this computation for free. As one can see in Figures 1(b-d) we typically consider input(s) to our physical body in form of forces or torques. Either the environment (or some other agent) applies them or they come directly from an active degree of freedom, i.e., any type of actuation, within the robot. The body does not "know" that it is part of a computational device. It simply obeys the laws of physics and it reacts accordingly. Note that this also implies that the body does not over- nor under-compensate, since it is a stable physical system.[3] The proposed setup simply adds some linear readouts to the body to complete the computation. The body would react exactly the same, if there were no readouts at all. If this output is used, e.g., in a feedback loop as a control signal for the robot, the behavior of the robot of course should be different if we close the loop by adding the readout. However, the

[3]For example, a linear mass-spring-damper system reacts with a force proportional to the perturbation - not more, nor less.

body still does not "know" that it is part of this computational loop. This raises the philosophical question, sometimes referred to as the teleological principle, whether the body is carrying out computation at all. Although, we are not trying to answer this question here,[4] we can argue that Physical RC can still be of practical value, if it is able to provide a beneficial mapping of input streams onto output streams with the help of the physical body.

Another interesting implication of this property of "ignorance" of the body (i.e., it behaves the same, whether we add readouts or not) is that we can employ the same body for multiple computations on the same input at the same time. We simply have to add the according number of readouts. We refer to that as *multitasking* and this has previously been demonstrated in simulations in [13] and [14] as well on real platforms, see [22].

The Power of Linear Regression

As pointed out in the introduction the RC setup needs on one hand a reservoir, which is basically a complex nonlinear, dynamical system with fixed parameters for the dynamics, and on the other hand a linear, static readout, which is adapted during the learning procedure. As a consequence, the task to learn to emulate complex, nonlinear operators/dynamical systems can be, with the help of the reservoir, reduced to the much simpler task to find some linear output weights, i.e., to carry out linear regression. It has been argued in [13] and [14] that if learning was successful, all the nonlinearity and memory that is needed for the emulation of the given computational task can be considered to be outsourced to the physical body in such a setup[5]. As also discussed in [13] this fact implies a number of remarkable properties.

First of all, linear regression is fast and it is even guaranteed to find a global optimum. Furthermore, linear regression "picks" (i.e., assigns bigger weights) to signals that are more relevant to produce the desired output, i.e., it naturally deprecates irrelevant information. Linear regression also provides the possibility to employ online learning, e.g., Recursive Least Square (RLS) and others. Caluwaerts et al. demonstrated this possibility in their simulation work with tensegrity robots, see [5], where the structure learned online to improve its locomotion. Besides these nice properties linear regression has also a more hidden advantage. It averages over conflicting information. This feature is especially important when a feedback setup (as presented in [14] - see also Figure 1a) is used. In general, such a feedback setup is needed to emulate more complex computations, like stable nonlinear limit cycle (e.g., for locomotion) or dynamical systems with bifurcation behavior or with multiple equilibrium points (i.e., analog finite state switching machines). The process to learn in such a case takes place in open loop, i.e., the optimal feedback (the target output) is fed back to the body and the internal signals

[4] For an excellent discussion of this question we refer to the submission of Hoffmann and Müller in this e-book.

[5] The argument is that the readout is only linear and does not have any notion of memory (i.e, it is static). Hence, in such a setup nonlinearity and memory can only be contributed by the physical body.

(sensory information) are collected to calculate the optimal weights. In simulations, if we only use this data, we are able to learn the (almost) perfect trajectory of a given target limit cycle. However, as soon as we close the loop already numerical imprecisions drive the system away from this trajectory. It is not stable. The system is not robust. However, if we superimpose noise to the learning data, linear regression averages over a region in the state space (i.e., some space around the nominal trajectory) and inherently learns a region of attraction, which results in a stable limit cycle. The conclusion is that noise is crucial for the robustness in a closed loop setup. Remarkably, for example, in the octopus arm setup of [22] the necessary noise to learn a robust limit cycle is provided by the physical environment (e.g., from the sensors or any other noise of the physical body and the environment).

Another aspect of robustness can be observed when we use linear regression in its online version. If, for example, a sensor is broken (e.g., producing only a zero value or only noise), its value will be simply deprecated over time by the online learning approach, i.e., it will receive a decreasing weight. Furthermore, we speculate that even in the case when any other part of the body is broken (e.g., a spring, a motor) the setup is potentially highly resilient as well. Note that in this context one could use also more sophisticated online learning approaches to take advantage of the Physical RC setup. For example, one could employ reinforcement learning techniques (see [30] for an overview) that try to improve a reasonable working trajectory based on a given performance measurement (e.g., speed in locomotion). Another possibility is to combine it with learning techniques that are information theory driven, as for example used in [21]. Such approaches are especially useful if we don't have a desired target function and the system has to learn to exploit its "eigen-dynamics."

Finally, we want to point out that linear regression is also inherently unbiased with respect to what kind of sensor types (e.g., gyroscope, pressure, force, visual, etc.) are used. It is also able to deal with different scales, changing the number of sensors being used, or redundant information.[6] Interestingly, biological systems exhibit a huge number of internal sensors and, hopefully, we will see a growth in the number of sensors in robots in the near future as well.

The Limitations of the Physical World

A big difference between the classical RC approach and the use of real physical bodies are mechanical constraints. While in the machine learning setup you can virtually choose any parameters values to fit your task, when you use a real physical platform you have to deal with given limitations.

The first one is the constraint related to frequency. A (stable) physical system works as a (nonlinear) low-pass filter. Mechanical parts cannot follow arbitrary frequencies and might be able to respond only at lower frequencies.

The second constraint (when compared to the virtual RC approach) is the transfer and distribution of information. In the standard RC approach, typically, all nodes are

[6] Clearly, if all signals are linearly dependent, we don't get the necessary computational power.

allowed to be connected with any other node.[7] However, in a real physical systems the information has to travel through the body by mechanical interaction. For example, if the octopus robot arm of Figure 1c is excited at the shoulder (on top in the depicted platform) than it will take some time until this information is carried along the passive silicone structure to have some effect on the tip of the arm. Although this seems to look like a disadvantage, after all we discuss here constraints, this can also be the right amount of fading memory that is needed for this robotic structure to be useful in certain computational tasks. This points to the fact that it makes sense to consider the computational role of physical bodies already during the design process.

The third constraint we discuss is noise. This could be introduced by physical effects (e.g., sensory noise) or by numerical imprecision, when the involved signals are digitalized. While in general noise is perceived as a disadvantage, in the case of the RC setup for morphological computation, as already pointed out, it can be an advantage. In the case of a learning setup, where we want to use the body in a closed loop (Figure 1a), noise makes sure to provide a broad enough set of learning data points around the desired trajectory to get a robust closed loop system. Of course the amplitude of the noise is crucial. Remarkably, in our experiments with real physical platforms we have not encountered any problems with respect to that so far. It seems the existing noise in the system, e.g., from the sensor and/or the body itself or the environment, provides the right amount and type of noise. In the context of noise we would also like to point the interested reader to the work of Hänggi [11], where the author demonstrated the usefulness of noise in biological systems in other context.

In summary, it is important to consider physical constraints present in robotic platforms. To be more specific, we argue that is important to consider them already during the design process as they directly influence the computational power of the physical body. Although, constraints constitute a problem and they clearly point to limitations of morphological computation, they can also help us to decide which part should be or can be outsourced to the physical layer and which part should be retained in the more classical electronic controlling units.

The "Body" and the "Brain"

In this section, we follow up on the question what should be or what can be outsourced to the morphology of the robot and which part should be implemented in the abstract, digital layer of the CPUs. Where should we draw the line between morphological computation and pure digital computation?

Before we do that we have to clarify what we mean by morphology (physical body) and digital layer (controller) in this context. Clearly, any virtual controller or digital controlling unit needs a physical embodiment. This could be, for example, a simple CPU, a complex signal processor, or even a full-blown computer. However, when we

[7] The Liquid State Machine approach [20] limits that by allowing some Gaussian probability for the connections, i.e., nodes are more probable to be connected to closer nodes than to ones that are farther away. This is done in order to mimic the connectivity pattern in the brain.

talk about morphology we are explicitly not considering classical computational architectures. The reason is that in these cases the physical implementation is designed to be as independent from the physical realization (i.e., the morphology) as possible. This is exactly the opposite of what we consider to be useful in morphological computation. A computer architecture is meant to be robust against any external influences, which are not defined as computational (digital) input. Loosely speaking, we don't want any bit to flip because an agent applies forces to the physical body of the computer, nor should any environmental change (like increasing temperature) influence the outcome of the computation.[8] On the other hand, in the morphological computation setup input is received directly through the body by interaction with the environment including other agents in form of forces. The robot body serves as actuator (output - it applies forces) as well as sensor (react physically to input forces). Another difference is that classical computational hardware is based on digital information (because it makes it more robust against external influences), while morphological computation lives in the continuous realm – it is analog computation. For a further and more in-depth discussion we refer to [10].

Coming back to the notion of morphology, when we talk about a physical body or morphology we refer to the part that is involved in morphological computation. When we use the term controller, or virtual or abstract computation we refer to the part that is implemented in form of classical computational architectures (e.g., a software program in a CPU). Note that the abstract (second) layer (in macroscopic biological systems these are neural networks) is typically on top of the morphological computation layer. Another important point is that classical robot designs (built out of rigid body parts connected with high torque servos) don't have any deliberate computation in the physical layer, hence, any computation in the robot is carried out only in the abstract controller layer.

Having clarified the terms we can now assess what kind of computations should be outsourced to the morphological layer and which computation should remain in the domain of the abstract control level. Based on the discussion of the previous section on constraints we can argue that it makes sense to outsource only computations that are in the frequency range inherent to the physical body. This would allow us to take advantage of this analog domain, which does not suffer from errors and delays introduced by digitalization processes. Hence, such computation is extremely fast. This is especially of advantage in computations that don't need to consider the long time history of inputs nor the integration of multiple sensory information. Reflexes are an excellent example for potential applications. Another typical field of application of morphological computation is given in tasks, where physical interaction with the environment or another agent is predominant. There is no need to artificially transfer the sensory information to a central processor, as it can be "used" directly and locally in a morphological computation setup. For example, legged locomotion is based on the appropriate interaction of the legs with the ground. But also efficient flying and swimming depend on the "appropriate" physical interaction with the engulfing medium. Grasping is another example where

[8] The only possibility is if we define a corresponding digital input to the computer as, for example, a thermal sensor.

morphological computation seems to be especially suited.

On the other hand, considering the given limitations of the physicality we can also conclude that more complex computational tasks, like planning or complex decision making, should be realized in the digital domain, as we can take advantage of features of classical computer architecture like long-term memory, look-up tables, and others.

Of course, there is still a gray area, i.e., computational tasks that can either be implemented in the morphology or in the digital domain. Hauser et al. showed in [13] and [14] that theoretically there are almost no limitations to outsourcing computation to the morphological layer. However, there are of course practical limitations as already pointed out. We would like to argue here again that these limitations are a good starting point for further investigations to understand better, where to draw the line. We also suggest that a feasible way towards a more general understanding is to look at specific tasks and to build corresponding specific morphological computation based robotic devices.

When outsourcing computation to morphology we have to consider another important point. While on one hand this can be a very elegant solution, on the other hand, by doing so we fix the type of computation it can carry out. We loose the ability to be adaptive. One possibility to avoid the necessity of adaptiveness at all is to design the physical body in such a way that it already exhibits inherent robustness over a wide range. For example, a mass-spring-damper system can move back to its resting length after a perturbing force has vanished. In a dynamical system terms, we talk here about a region of attraction around a point of stability. This attractor space can be shaped by the mechanical design and is defined by the physical properties. Accordingly, for the case of repeating patterns, we can consider stable limit cycles with corresponding attractor regions around the nominal trajectory in the state space. However, there will be physical limitations for such attractor regions. Furthermore, for certain task qualitatively different responses are required, e.g., a switch from a repetitive movement to a goal driven movement (i.e., limit cycle vs. equilibrium point).

A better possibility and a more general approach to overcome the problem of the missing adaptivity is to include some adaptation mechanism to change the morphology online. We call that *morphosis*. Such a change can include the adaptation of single parameters (e.g., the stiffness, friction coefficient, etc.), but also geometrical arrangements of body parts [26], actuators [33, 31], or sensors [19].

The EU project LOCOMORPH has investigated that line of research in the context of locomotion and built a number of novel morphosis mechanisms. The results also demonstrate how such changes can be beneficial with respect to energy efficiency and how they can increase the region of sensible parameter space over different terrain properties [32]. Typically, such changes take place on a slower time scale than the actual movement and, important in the context of this discussion, such changes are then meant to be initiated by a controller at a higher level, while at the lower level the morphological properties assure a stable movement.

Another aspect of the line between abstract controller layer and mechanical body is the information flow between them. One possibility is to use the body as some nonlinear preprocessing devise to transform information in such form that the controller can deal more easily with it. An often cited example from biology is the insect eye, where the

individual rather simple visual sensors are arranged in such a nonlinear way that it counteracts (i.e., linearizing) the nonlinear effect of the optical flow, see [9]. There exists also a corresponding implementation of this principle in a robotic platform, see [19]. One could say that the morphology linearizes the input data stream. Of course, the preprocessing in the case of the insect eye is, although being nonlinear, only a simple static mapping. However, in the context of morphological computation (specifically in the case of Physical RC) a physical body can be used for much more complex tasks. For example, it could be used for preprocessing tasks that include time depending computations, i.e., the use of memory is required. This has been demonstrated in simulation results in [13], but was also shown to work in real physical platforms, see, e.g., the octopus experiments on [22]. Besides the information flow from the body to the abstract controller layer, there is also a flow into the other direction. The idea is that the high-level controller (brain) only interferes when necessary. One example has already been pointed out when we discussed the concept of morphosis, where the high level controller initiates the switch to a different behavior by adapting the morphology to the most appropriate one for the current situation (i.e., computational task). In the following section, we will elaborate this topic in more details.

Switching Limit Cycles - Switching Behavior

Füchslin et al. discussed in [10] the idea of morphological control, i.e., computation carried out for the sake of control with the help of morphology. They described a setup where the physical part defines either a stable limit cycle or a point of equilibrium with its corresponding area of attraction. This means if the (physical) system is perturbed (in a certain maximal range) it finds its way back to the nominal behavior (either limit cycle or point of equilibrium). Now, for a different task the system might need to embody another behavior. It has to switch, for example, from one limit cycle to another, or even switch from a point of equilibrium to a limit cycle (bifurcation). Here is where the abstract control layer comes into play. It does not have to fully control the whole body, it just has to provide the appropriate input to move the system out of the actual region of attraction (of a limit cycle or equilibrium point) into a new one. One could also imaging to have a mechanism that changes the morphology of the system itself (morphosis mechanism as previously discussed). One could move a certain link of the body into a different posture. Füchslin et al. give the example of humans who change their body posture when they have to carry a heavy backpack. Another example is the "kick" we have to give our body when we change our gait patterns, e.g. to change from walking to running.

In [14] Hauser et al. demonstrated such a switch in their simulations. The input to the physical body (serving as a reservoir) was a constant force to randomly chosen locations[9] in the body. Depending on the amplitude of the input force the system produced autonomously one out of three different limit cycles. Note that, since the output weights are fixed after learning and we apply forces as input to a compliant body (reservoir), we

[9] These locations are randomly chosen in the initialization phase, when the reservoir is constructed as well. However, these locations are fixed afterwards.

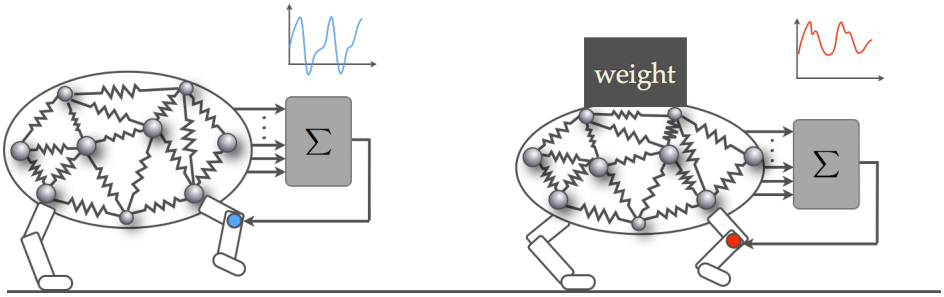


Figure 2: *Schematics of a possible setup to sense different environmental conditions through a soft body that serves as a reservoir. For both situations the linear setup is kept constant. On the left side the system produces autonomously a limit cycle that serves as a control signal for one degree of freedom (e.g., for the knee motor). On the right side a heavy weight is put on the body (the environment has changed) and the state of the reservoir changes. As a consequence, the system produces now a different control signal for the knee.*

actually change the behavior of the system (i.e., different limit cycles) by simply squeezing the body. If we squeeze it hard it produces a different limit cycle than if we squeeze it a little.

One could now consider, for example, a legged robot with such a compliant body (compare Figure 2). The Physical RC setup should produce the motor position for one degree of freedom (a limit cycle for a steady locomotion) by reading out the state of the body (via internal sensors). Now, if we add a heavy weight to the back of the robot, the system should produce a different limit cycle, e.g., now for a gait with the knees more bent which is more appropriate for carrying this weight. It is important to note here that the output weights do not have to be changed. Just the different environmental conditions (heavy weight on the robot vs. the robot's weight alone) would initiated the change. One could argue the body is able to sense this difference, which is a remarkable conclusion, as this is a notion completely undiscussed in classical robot design literature. In the next section we will elaborate more on this idea of sensing through the body.

However, before we do so, we would like to show this idea of switching behavior can be carried even further. Consider you have a couple of limit cycles that serve as controller trajectories for a number of active degrees of freedoms, for example, for a humanoid robot to locomote. Now if the robot stumbles and falls, it would need a completely different strategy to get up again. Instead of limit cycles, the robot would need for its degrees of freedom appropriate trajectories in its high dimensional space towards points of equilibrium (which correspond to the robot standing). For more complex robots, one could also consider some intermediate sets of equilibrium points - some sort of waypoints, which define subgoals on the way to stand up. After that the robot needs again a strategy to move from this rather static posture back into the limit cycles of walking (bifurcation). The transitions between these completely different dynamic behaviors will be initiated by a high level controller in the previously discussed abstract controller layer. Note that the controller only has to initiate the transition, i.e., give it enough push (into the

right "direction") to get out of the region of attraction. Note that this control signal in general does not have to be very precise as the rest of the stable movement is "encoded" in the morphology. This also means that the controller only needs to know in which basin of attraction the system presently resides to initiate the jump into another one. Consequently, it only needs to know (and measure) the approximate state of the system. In the context of Physical RC, assuming the body is designed cleverly enough, simple (smooth) switches between different sets of output weights to produce the actuator signals might be sufficient to guide the robot through all these stages.

The Smart Body - Sensing Through the Body

As mentioned in the previous section, the body can be used as some kind of complex sensor. This goes along with the idea presented in the section "The 'Body' and the 'Brain'" where we outlined the idea of using the body as a computational preprocessing unit.

While in the machine learning setup of RC the input is some abstract signal, in a real physical systems the inputs have underlaying physical effects. For example, in the simulations presented in [13] and [14], which intent to simulate real physical systems, inputs are defined as forces. Since the physical reservoir is compliant it reacts to these forces and changes its state accordingly.

Let's take a closer look at what kind of forces are considered and where they potentially can come from. For example, forces can be applied by internal actuators. This was the case in the octopus experiments in [24, 23, 22], see Figure 1c, where an attached motor moved the passive soft arm structure. In the quadruped experiments of Zhao et al. [34] the locomotion motor applied forces to the body and, therefore, introduced changes in the soft spine, which served as a physical reservoir (see Figure 1d).

The input forces could also come from the environment as in the case of the "heavy weight" example, see Figure 2. Due to gravity the additional weight applies some forces to the soft body and, therefore, changes the state of the physical reservoir.

Another possible way to receive input forces from the environement are external objects, which, e.g., don't allow the standard locomotion limit cycle to succeed (i.e., the robot hits an obstacle with one of its legs). Moreover, already subtle changes in the ground friction can introduce forces during locomotion, which could be then "sensed" through the body. Again, a soft body (the reservoir) would deform accordingly. A similar effect has been used in the work by Owaki et al. [25]. They showed that a mechanical communication (via the body as well via the environment) between limbs seems to be essential for robust quadruped locomotion. Furthermore, Caluwaerts et al. showed in [5] exactly the discussed sensing capabilities with a simulated tensegrity robot. They demonstrated that their robot is able to learn to distinguish different environmental conditions (flat ground, ground with nobs) by processing it through the compliant tensegrity structure during locomotion.

Another input possibility are forces that are introduced by an external agent, e.g., another robot or a human guiding the passive robot. Finally, forces as input could also

be provided by an object that a soft robot wants to grasp. There would be an immediate force feedback, which could be, for example, used to differentiate various objects.

From all the examples we can immediately see that sensing through a physical body is possible without the use of specific sensors. It is a very direct and fast sensing approach and no external artificial control loops are needed. However, it must be also clear that there is certain sensory information that is not directly accessible through a force interaction, for example, the information on radiation or visual input, just to name two. Nevertheless, as previously pointed out in the section on "The Power of Linear Regression" the Physical RC approach is capable to incorporate such sensory information without any problem when provided by an appropriate sensor.

Finally, we would like to point to the fact that sensing through the body needs a compliant body, as well a certain extent of passive degrees of freedom (i.e., an underactuated system). Both are typical properties of soft robotic structures. Hence, we argue that soft robots are especially well suited for the Physical RC approach.

Where Does the Reservoir Start and Where Does it End?

In this final section, we will discuss the question where does the reservoir start and where does it end? The answer might be trivial for the abstract machine learning setup, since in that case the reservoir is explicitly defined by the designer. However, it is not that simple to find an appropriate response in setups with real physical platforms. Let's take, for example, the octopus platform from Figure 1c. At the first sight one might say that the silicone structure is the reservoir. However, this is only partially true. It is right that the (input) forces applied by the motor are transformed into a change of the dynamical state of the arm. However, there is a significant part that is contributed by the dynamics of the interaction of the arm with the environment, i.e., the water. There are nonlinear effects, drags, damping effects and so on. If we would change the property of the liquid significantly (e.g., changes the density by adding salt) we would get different responses in the sensor outputs for the same motor input signal. Note that any nonlinear effect and any temporal integration can provide potentially additional computational power as they can add to the kernel property and to the fading memory as discussed in the introduction (see also [13, 14]).

To give another example, the reservoir of the quadruped robot of [34] in Figure 1d is not just the compliant spine. All other body parts contribute to the reservoir as well, since the feedback loop from the motor signal (produced by the Physical RC setup) goes through the environment, via the contact points at the feet, back to the rest of the body of the robot. As previously discussed a sufficiently big enough change in the environment would change the sensory values and, therefore, the behavior of the whole system.

Next to the contribution of the interaction with the environment, actuation systems and sensor parts of the robotic structure can provide beneficial nonlinear effects to the reservoir as well. Any type of actuator as well as sensors have typically some sort of nonlinearities, e.g., saturations, memory effects, etc. They are usually perceived as disadvantages. Interestingly, in the Physical RC setup they can potentially contribute to

the computational power.

Finally, when we use this setup to produce a control signal for the robot, remarkably, we can conclude that the body itself can be used as a computational resource to control itself. This means the classical separation between controller and the to-be-controlled is blurred and, therefore, we have to rethink what control means in this context. A start of this discussion has been laid out in [10].

Conclusion and Future Outlook

We have discussed a number of implications when the theoretical models of [13] and [14] are implemented in real physical platforms. There are a number of remarkable conclusions, since the involved parameters from the underlaying machine learning technique (i.e., reservoir computing) relate to real physical properties. We discussed the interpretation of physical bodies as computational resources and show how the complex task to learn to emulate relevant computations can, with the help of such a body, be reduced to simple linear regression. We also provided a number of remarkable consequences, when we can use such a simple learning setup. We pointed out physical constraints and proposed how we could deal with them. We investigated the role of the body in relation to abstract controller levels and showed how a compliant physical body can be used as some sort of "full-body sensor" and as a preprocessing unit.

We hope that the individual sections of this article will inspire people to do more research into that direction. We would like to refer the interested reader to the introduction of a special issue on morphological computation [15] where we have pointed out a number of research opportunities, which are also relevant for this specific Physical RC approach.

Furthermore, we believe the future holds a number of interesting directions for this approach. As the research on growing and self-assembling artificial material grows more mature we will be able to build a whole range of interesting physical bodies. There exists already some theoretical work as well some simulations that show how physical bodies can be optimized to increase the performance for given tasks [16]. Also the work by Bernhardsgrütter et al. [2], where L-systems are used to control the grow of a physical reservoir, show promising results.[10] If we are able to guide the growth or assembly of such real physical structures we would have a wonderful tool at our hand to build computationally powerful bodies.

Although we have mostly discussed examples from robotics, the same principles can be employed for a wider range of intelligent bodies. One could apply the same framework, for example, for smart furnitures or building structures, e.g., a chair or a floor that is able to distinguish how and who is interacting with it. Another possible way to exploit the morphological computation setup is to consider smart materials, which can change their dynamic properties by applying electric or magnetic fields, or by controlling the temperature. Such physical structures would be more adaptive and resilient. For example, they could change their resonance and filtering behavior when appropriate.

[10] Their code is freely available on https://github.com/SoftRobotics.

So far, the discussed applications and implementations almost exclusively focused on systems governed by classical mechanics and/or continuum mechanics. However, the underlying principles apply as well to systems with dynamics governed by statistical mechanics. One important aspect of this notion is the hypothesis that also cellular control is not only enforced by genetic signals but embodies the morphology of cellular components such as, e.g., membranes as active parts in information processing. Taking the body, on the cellular or on the macroscopic level, not only as the stage on which control processes take place but as an active part of the biological computational infrastructure will influence the way how we design therapies.

In summary, due to the generality of the physical reservoir computing approach a broad range of applications are possible and we believe especially an interdisciplinary approach will be highly beneficial in this context. We are excited to see what people are coming up with.

Acknowledgment

The project has been partially supported by the EU project FP7-PEOPLE-2013-ITN 608022 SMART-E. Helmut Hauser would like to thank Fumiya Iida for the initial idea to consider switching between more complex movements in the attractor space.

Bibliography

[1] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer. Information processing using a single dynamical node as complex system. *Nature Communications*, 2:468, 2011.

[2] R. Bernhardsgrütter, C. W. Senn, R. M. Füchslin, C. Jaeger, K. Nakajima, and 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)*, 2014.

[3] S. Boyd. *Volterra Series: Engineering Fundamentals*. PhD thesis, UC Berkeley, 1985.

[4] S. Boyd and L. Chua. Fading memory and the problem of approximating nonlinear operators with volterra series. *Circuits and Systems, IEEE Transactions on*, 32(11): 1150–1161, Nov 1985. ISSN 0098-4094.

[5] K. Caluwaerts, M. D'Haene, D. Verstraeten, and B. Schrauwen. Locomotion without a brain: physical reservoir computing in tensegrity structures. *Artificial Life*, 19: 35–66, 2013. ISSN 1064-5462. doi: 10.1162/ARTL_a_00080.

[6] K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral. Design and control of compliant tensegrity robots through simulation

and hardware validation. *Journal of the Royal Society Interface*, 11(98):20140520, 2014.

[7] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar. All-optical reservoir computing. *Optics Express*, 20(20):22783–22795, 2012.

[8] C. Fernando and S. Sojakka. Pattern recognition in a bucket. In *Advances in Artificial Life*, volume 2801 of *Lecture Notes in Computer Science*, pages 588–597. Springer Berlin / Heidelberg, 2003.

[9] N. Franceschini, J. M. Pichon, C. Blanes, and J. M. Brady. From Insect Vision to Robot Vision [and Discussion]. *Philosophical Transactions Biological Sciences*, 337 (1281):283–294, 1992. ISSN 09628436.

[10] R. M. Füchslin, A. Dzyakanchuk, D. Flumini, H. Hauser, K. J. Hunt, R. H. Luchsinger, B. Reller, S. Scheidegger, and R. Walker. Morphological computation and morphological control: steps toward a formal theory and applications. *Artificial Life*, 19(1):9–34, 2013. ISSN 1064-5462. doi: 10.1162/ARTL_a_00079.

[11] P. Hänggi. Stochastic resonance in biology. *ChemPhysChem*, 3(3):285–290, March 2002. doi: 10.1002/1439-7641(20020315)3:3<285::AID-CPHC285>3.0.CO;2-A.

[12] H. Hauser and G. Griesbacher. Moving a robot arm by exploiting its complex compliant morphology. In R. Pfeifer, S. Hidenobu, R. M. Füchslin, H. Hauser, K. Nakajima, and S. Miyashita, editors, *Proceedings of the 2nd International Conference on Morphological Computation*, September 2011.

[13] H. Hauser, A. Ijspeert, R. Füchslin, R. Pfeifer, and W. Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105:355–370, January 2011. ISSN 0340-1200. doi: 10.1007/s00422-012-0471-0. Issue 5.

[14] H. Hauser, A. Ijspeert, R. Füchslin, R. Pfeifer, and W. Maass. The role of feedback in morphological computation with compliant bodies. *Biological Cybernetics*, 106 (10):1–19, 2012. ISSN 0340-1200. doi: 10.1007/s00422-012-0516-4.

[15] H. Hauser, H. Sumioka, R. M. Füchslin, and R. Pfeifer. Introduction to the special issue on morphological computation. *Artificial Life*, 19(1):1–8, 2013.

[16] M. Hermans, B. Schrauwen, P. Bienstman, and J. Dambre. Automated Design of Complex Dynamic Systems. *PLoS ONE*, 9:e86696, 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0086696.

[17] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag GmbH, third edition, 2001. ISBN: 3-540-19916-0.

[18] H. Jaeger. Adaptive nonlinear system identification with echo state networks. in S. T. S. Becker & K. Obermayer, eds, 'Advances in Neural Information Processing Systems 15', MIT Press, Cambridge, MA, pp. 593–600., 2003.

[19] L. Lichtensteiger and P. Eggenberger. Evolving the morphology of a compound eye on a robot. *1999 Third European Workshop on Advanced Mobile Robots (Eurobot'99). Proceedings (Cat. No.99EX355)*, 1999. doi: 10.1109/EURBOT.1999.827631.

[20] W. Maass, T. Natschlaeger, 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.

[21] G. Martius, L. Jahn, H. Hauser, and V. Hafner. Self-exploration of the Stumpy robot with predictive information maximization. In A. del Pobil, E. Chinellato, E. Martinez-Martin, J. Hallam, E. Cervera, and A. Morales, editors, *From Animals to Animats 13*, volume 8575 of *Lecture Notes in Computer Science*, pages 32–42. Springer International Publishing, 2014. ISBN 978-3-319-08863-1. doi: 10.1007/978-3-319-08864-8_4.

[22] K. Nakajima, T. Li, H. Hauser, and R. Pfeifer. Exploiting short-term memory in soft body dynamics as a computational resource. *Journal of the Royal Society Interface*, 11(100), November 2014. doi: 10.1098/âĂŃrsif.2014.0437.

[23] K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer. A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. *Frontiers in Computational Neuroscience*, 7:91, 2013. ISSN 1662-5188. doi: 10.3389/fncom.2013.00091.

[24] K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer. Computing with a muscular-hydrostat system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1504–1511. IEEE, 2013.

[25] D. Owaki, T. Kano, K. Nagasawa, A. Tero, and A. Ishiguro. Simple robot suggests physical interlimb communication is essential for quadruped walking. *Journal of The Royal Society Interface*, 10(78):20120669, 2013.

[26] H. V. Quy, G. Ramstein, F. Casanova, L. Aryananda, M. Hoffmann, F. I. Sheikh, and H. Hauser. Gait versatility through morphological changes in a new quadruped robot. In *5th International Symposium on Adaptive Motion of Animals and Machines*, 2011.

[27] B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European Symposium on Artificial Neural Networks*, pages 471–482, 2007.

[28] A. Smerieri, F. Duport, Y. Paquot, B. Schrauwen, M. Haelterman, and S. Massar. Analog readout for optical reservoir computers. In *Advances in Neural Information Processing Systems*, pages 944–952, 2012.

[29] H. Sumioka, H. Hauser, and R. Pfeifer. Computation with mechanically coupled springs for compliant robots. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, September 2011.

[30] R. S. Sutton and A. G. Barto. *Reinforcement Learning - An Introduction*. The MIT Press, first edition, 1998.

[31] B. Vanderborght, A. Albu-Schaeffer, A. Bicchi, E. Burdet, D. G. Caldwell, R. Carloni, M. Catalano, O. Eiberger, W. Friedl, G. Ganesh, M. Garabini, M. Grebenstein, G. Grioli, S. Haddadin, H. Hoppner, A. Jafari, M. Laffranchi, D. Lefeber, F. Petit, S. Stramigioli, N. Tsagarakis, M. Van Damme, R. Van Ham, L. C. Visser, and S. Wolf. Variable impedance actuators: A review. *Robotics and Autonomous Systems*, 61:1601–1614, 2013. ISSN 09218890. doi: 10.1016/j.robot.2013.06.009.

[32] H. Vu Quy, H. Hauser, D. Leach, and R. Pfeifer. A variable stiffness mechanism for improving energy efficiency of a planar single-legged hopping robot. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–7, Nov 2013. doi: 10.1109/ICAR.2013.6766488.

[33] H. Vu Quy, L. Aryananda, F. I. Sheikh, F. Casanova, and R. Pfeifer. A novel mechanism for varying stiffness via changing transmission angle. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5076–5081, 2011. ISBN 9781612843865. doi: 10.1109/ICRA.2011.5980097.

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