On developing theory of reservoir computing for sensing applications: the state weaving environment echo tracker (SWEET) algorithm

INTERNATIONAL JOURNAL OF PARALLEL, EMERGENT AND DISTRIBUTED SYSTEMS, 2018
VOL. 33, NO. 2, 121–143
https://doi.org/10.1080/17445760.2016.1241880

Taylor & Francis
Taylor & Francis Group

On developing theory of reservoir computing for sensing applications: the state weaving environment echo tracker (SWEET) algorithm

Zoran Konkoli

Department of Microtechnology and nanoscience - MC2, Chalmers University of Technology, Gothenburg, Sweden

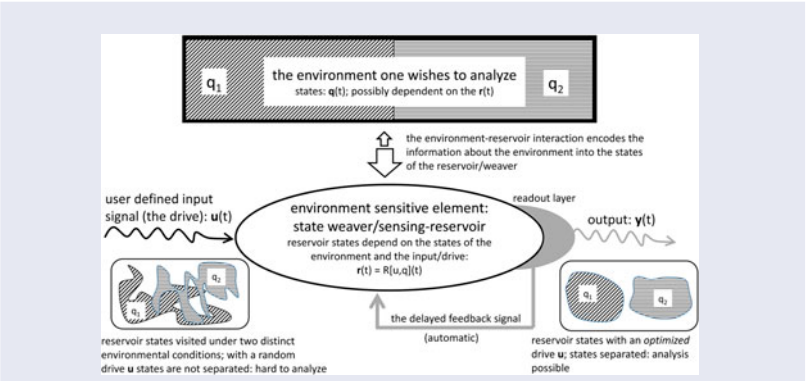
ABSTRACT

As a paradigm of computation, reservoir computing has gained an enormous momentum. In principle, any sufficiently complex dynamical system equipped with a readout layer can be used for any computation. This can be achieved by only adjusting the readout layer. Owning to this inherent flexibility of implementation, new applications of reservoir computing are being reported at a constant rate. However, relatively few studies focus on sensing, and in the ones that do, the reservoir is often exploited in a somewhat passive manner. The reservoir is used to post-process the signal from sensing elements that are placed separately, and the reservoir could be replaced by other information processing system without loss of functionality of the sensor ('reservoir computing *and* sensing'). An entirely different novel class of sensing approaches is being suggested, to be referred to as 'reservoir computing for sensing', where the reservoir plays a central role. In the State Weaving Environment Echo Tracker (SWEET) sensing approach, the reservoir functions as the sensing element if the dynamical states of the reservoir and the environment one wishes to analyze are strongly interwoven. Some distinct characteristics of reservoir computing (in particular the separability and the echo state properties) are carefully exploited to achieve sensing functionality. The SWEET approach is formulated both as a generic device setup, and as an abstract mathematical algorithm. This algorithmic template could be used to develop a theory (or a class of theories) of 'reservoir computing for sensing', which could provide guidelines for engineering novel sensing applications. It could also provide ideas for a creative recycling of the existing sensing solutions. For example, the Horizon 2020 project RECORD-IT (Reservoir Computing with Real-time Data for future IT) exploits the SWEET sensing algorithm for ion detection. Accordingly, the terms SWEET sensing algorithm and the RECORD-IT sensing algorithm can be used interchangeably.

ARTICLE HISTORY
Received 2 September 2016
Accepted 23 September 2016

KEYWORDS
Reservoir computing; sensing; the SWEET sensing setup; algorithmic template

CONTACT Zoran Konkoli zorank@chalmers.se



1. Introduction

In the last decade a relatively large number of studies have reported information processing applications where reservoir computing features as the central component. The number of such applications is increasing at a constant rate. The paradigm of reservoir computing has been a popular method of choice in a range of information processing applications, and various types of reservoirs have been explored. Instead of listing specific applications the reader is directed towards a few recent reviews.[1–5]

A reservoir computing setup features two key elements, a dynamical system that can respond to inputs, a reservoir, and a readout layer that is used to analyze the state of the system. To be used this way, the reservoir should have a set of rather generic properties, the most important ones being the echo state property and the input separation property. In somewhat simplified terms, the reservoir transforms the input into the internal states of the dynamical systems, and to compute unambiguously, different inputs need to be mapped to different states. The echo state property ensures that the initial state of the reservoir does not influence the result of the computation, which allows for on-line computation. The key insight regarding this type of information processing is that the transformation from the input into the internal state of the system is in fact the computation performed by the system. The readout layer is used only to access the result of the desired computation. It should possess the approximation property, i.e. be able to approximate any multi-variable function (on the set of states) to any desired degree of accuracy. However, if the reservoir *per se* is complex enough, then the readout layer can be very simple, e.g. even a linear readout might suffice.

Thus the reservoir computing paradigm is extremely flexible, and relatively straight forward to implement. In practice, recurrent neural networks have been used as reservoirs, but this need not be the case, other systems could be easily exploited, just that they have the required generic properties. Because of this, the reservoir computing paradigm harbors an enormous application potential. A more detailed discussion regarding such technological promise of reservoir computing (and its limitations) can be found in [6]. Despite its obvious appeal, reservoir computing has not been extensively explored in the sensing context. The goal is to perform a generic analysis of possible uses of reservoir computing for realizing sensing applications. In particular, one can speculate whether it is possible to envision novel sensing setups where the reservoir is used actively as the sensing unit.

The overall goal of this study is to critically reflect on possible uses of reservoir computing for sensing with the following *two big questions* in mind (and the related sub-questions):

(Q1) *Are there specific characteristics of reservoir computing that make it particularly useful for designing novel sensing strategies?* Can we develop a generic theoretical framework that can describe such strategies in rigorous mathematical terms?

(Q2) *Are there specific sensing applications that could be easily engineered using reservoir computing?* Given a sensing problem, which reservoir-computing-based sensing strategies would be particularly useful, desirable, and, perhaps, even necessary to solve it?

The scope of the above questions is rather broad and addressing them would clearly be out of the scope of a single study. Some specific possibilities are addressed instead. In particular, the inherent flexibility of the reservoir computing paradigm (that likely contributed to its popularity) is a characteristic that one could try to leverage systematically when targeting sensing applications. For example, in situations when there are severe engineering constraints on how the sensing is to be done, the inherent flexibility of reservoir computing could be exploited to deal with the constraints: In biomedical applications there is a need to develop sensing procedures that are bio-compatible, where an embedded or low-power sensing needs to be done.

The questions (Q1) and (Q2) will be critically addressed with the following ideas in mind. First, an attempt will be made to formulate a generic theory of reservoir computing for sensing. Hopefully, the theory will provide a basis for a precise conceptual understanding of the problem. Second, the theory will be used to critically reflect on what needs to be done to engineer key theoretical concepts, and where the limitations of the approach might be.

Two types of sensing approaches are distinguished. First, the studies where reservoir computing appears as an afterthought rather than a central element in construction of the sensor will be referred to as 'reservoir computing and sensing'. Second, the question is whether one can envision applications in which reservoir computing plays a more prominent, perhaps even a central role. These approaches will be labeled as 'reservoir computing for sensing'.

If in doubt, the following thought experiment can be used to decide how to characterize an arbitrary sensor device were reservoir computing is used. If it is possible to replace the reservoir without significantly altering the functionality of the sensor, then such a system falls into the 'reservoir computing and sensing' category. In such a sensing setup the reservoir does not play a central role. Conversely, if the replacement of the reservoir is not possible (without altering the sensor functionality) then such a sensing system is classified as 'reservoir computing for sensing'. In such a setup, the reservoir computing concept features as a key element in the design of a sensor.

The investigations that specifically focus on the use of reservoir computing in the sensing context are still relatively few but some representative pioneering efforts can be found in [7–16]. These focus on vastly different tasks related to the problems of the robot control,[7–9,13,17,18] ambient sensing,[10,11], gas analysis [12,14] classification of optical signals,[15] or structural health monitoring (of a bridge).[16] Further, the paradigm of reservoir computing has been extensively used to analyze time series data, and there are numerous cases that might be classified as sensing too, perhaps in a broader sense of the word, such as the analysis of ECG electrode signal to detect a heart failure (*cf.* [19] and references therein).

The majority of such studies can be classified as *reservoir computing and sensing*. In these applications the sensing setup often exploits an artificial neural network to process an information that comes from a set of sensors. The readout layer is trained to characterize this information. However, reservoir computing is not actively used for sensing and features as an afterthought. Instead of reservoir computing techniques other algorithms could be used to train the network (e.g. a back-propagation method). Ultimately, the time series data that come from the sensing unit could be processed in a completely different way (e.g. by using some artificial intelligence system).

A small fraction of the reported applications can be classified as *reservoir computing for sensing*. For example, a few exciting applications have been reported in the context of morphological computation related to the soft-robot control,[8,9,13] where the mechanical deformation of the robot is exploited for information processing. There is obviously a lot of potential to generalize and explore new possibilities.

In here, a novel sensing paradigm will be presented, that falls into the 'reservoir computing for sensing' class, to be referred as the State Weaving Environment Echo Tracker (SWEET) paradigm,[20] and will be presented in the form of a conceptual device design setup, and a mathematical algorithm.

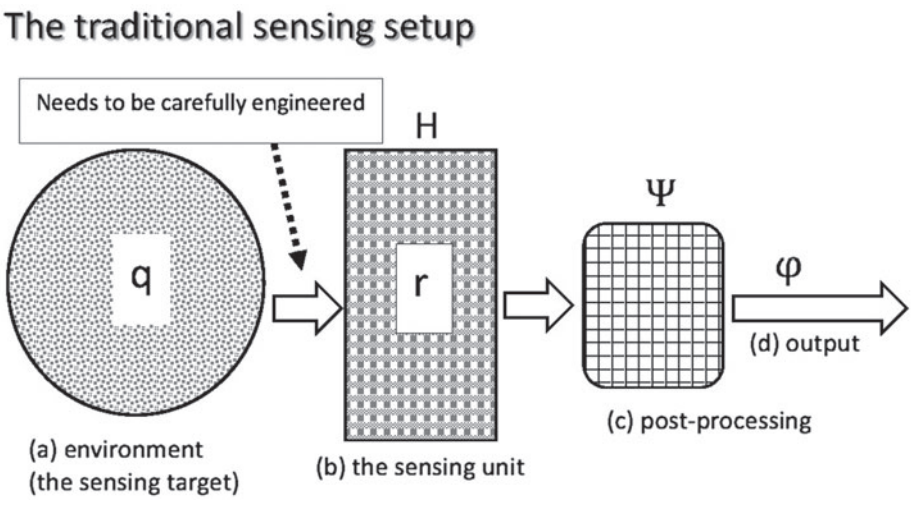


Figure 1. The traditional sensing setup: The sensing unit *H* is used to measure certain features of the environment one wishes to analyze. The features are denoted by *r*. The assessment is instantaneous. The extracted features are used to draw a final conclusion about the state of the environment through the (optional) post-processing step.

Both can be used as an aid in engineering novel sensing applications (but also for a creative recycling of the existing ones). For example, the SWEET algorithm could be used as a base to develop a theory of reservoir computing for sensing in specific applications domains (e.g. ion concentration analysis), and the abstract objects in the algorithm can be further engineered in the context of the SWEET device design.

The text is organized as follows. Section 2 describes the key ideas behind the SWEET sensing concept: its ingredients, and how they are expected to interact together. Section 3 discusses a minimal theory of reservoir computing for sensing. Both the traditional and the SWEET sensing setups are formalized in precise mathematical terms. Section 4 contains a discussion on the expressive power of the SWEET paradigm. A set of practical guidelines for implementing it are listed in Section 5. The concluding Section 6 contains a list of key features of the SWEET sensing setup and indicates some future research directions.

2. The SWEET sensing concept

In this section, the key ideas that the SWEET sensing concepts builds on are presented. The essential components are introduced and their functionality discussed. This is done in two steps. The ideas behind the traditional sensing approach are discussed first. This is followed by a discussion of the SWEET setup. In this way the abstract SWEET ideas are introduced gradually. Further, the two approaches are compared, which highlights the distinct features of the SWEET setup.[1]

Figure 1 depicts a schematic representation of the traditional sensing setup, which consists of the sensing unit and the environment one wishes to analyse, the sensing target. In this approach the interaction between the sensing unit and the sensing target has to be carefully engineered. Usually one needs to ensure that this interaction is strong, e.g. to overcome problems with noise. Further, the interaction has to be unidirectional, the sensing unit should not influence the sensing target, otherwise an erroneous information can be inferred about the sensing target.

In the novel sensing approach that is being suggested, the SWEET sensing concept, the reservoir is used more actively. Even subtle changes that the environment imprints into the reservoir are exploited to infer the state of the environment. In this setup the concept of time plays a crucial role since the time sensitive features of the reservoir dynamics are exploited to the largest possible extent. Figure 2 shows the schematics of the conceptual device design.

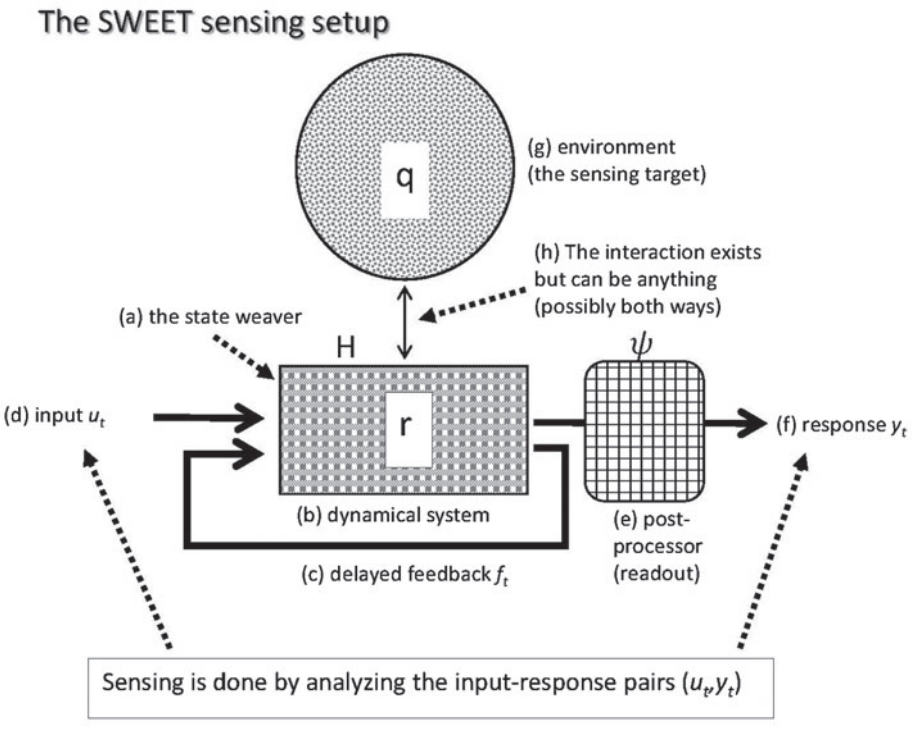


Figure 2. The SWEET sensing setup features the dynamical system *H* that interacts with the environment. The states of the dynamical system and the states of the environment are strongly entangled, so that the history of the environment influences controls the state of the system. The dynamical system functions as a weaver of the states of the environment traced over a longer time period. It is assumed that the weaver can be driven by two inputs, the external input *u*, and a delayed feedback *f*. The functionality of the delayed feedback is enhanced the degree of state entanglement. The setup is extremely flexible, as an indirect sensing procedure is used: the user provides an input *u*, and the response of the system *y* is used to infer about the state of the environment. Note that the user never needs to measure the internal degrees of freedom of the state weaver, *r*.

2.1. The key components

The setup features an intimate connection between the sensing equipment and the environment one wishes to analyze and consists of the following components:

- an environment one wishes to sense (the sensing target).
- an environment sensitive element (the state weaver or, equivalently, the sensing reservoir)
- a user defined input signal (the drive)
- the delayed feedback signal (the automated input)

The sensing procedure is based on an indirect sensing approach: the response of the sensing unit is studied under different environmental conditions, and possibly different drives. If different environmental conditions lead to radically different responses, one should be able to infer the state of the environment by inspecting how the device responds to different drives.

2.1.1. The super-reservoir concept

In the SWEET sensing approach, the sensing element and the environment are both treated conceptually as one dynamical system, a *super-reservoir* in which the states of the weaver and the environment are entangled. Owing to the echo state property, the state of the super-reservoir at a particular time instance embodies everything that both the weaver and the environment have experienced in the past, as a unit. However, ultimately, it is the states of the weaver one is interested in: the information

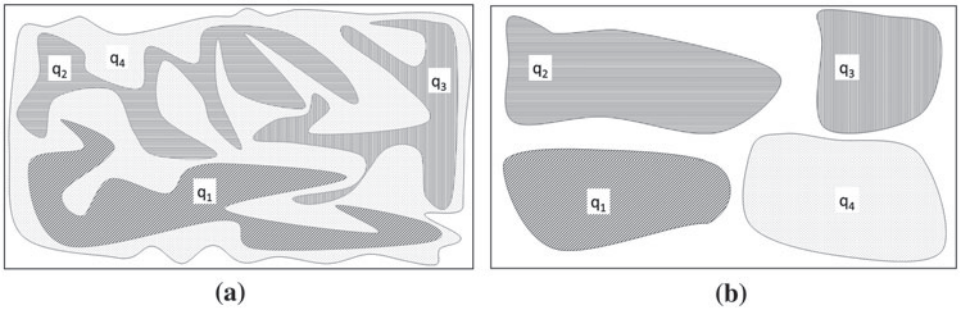


Figure 3. An illustration of the SWEET sensing idea. Panel (a): Under the influence of four environmental conditions q_1, q_2, q_3 and q_4 the system (reservoir) visits different regions of the configuration space. For example, if exposed to the environment q_1 the system visits only the points in the diagonally patterned region. It is hard to perform sensing since it would be challenging to construct a readout layer that could identify which region of the configuration space is being visited under an unknown environmental condition (e.g. region q_4 might be especially problematic since it is strongly interwoven with other regions). Panel (b): With an additional input, if such can be found, the phase space partitions. It is easier to construct a readout layer that can separate between the regions.

about the environment will be embedded (encoded) into the state of the weaver. If there is a way to analyze the weaver states, then it should be possible to infer indirectly about the environment.

2.1.2. Information processing

How can one extract the information about the environment that is stored in the state of the weaver? The purpose of the user defined input signal is to make this information available. This idea is illustrated in Fig. 3. If a suitable input can be found that separates the regions of the configuration space of the reservoir which are visited under different environmental conditions, then a relatively simple readout layer could be used to infer about the state of the environment. In fact, it will be shown later that the input signal (in combination with the readout layer) can perform three tasks simultaneously: decoding, extraction, and analysis (*cf.* Section 3.5.3).

2.1.3. Delayed feedback

The delayed feedback, or just the feedback, mechanism is extremely important. It is included for two reasons, to increase the richness of the configuration space of the super-reservoir, and to achieve synchronization between the environmental signal and drive u. There are several reasons why the delayed feedback can achieve this:

- It is exactly the presence of non-linear feedback loops in the neural network reservoir that makes the structure of the configuration space suitable for reservoir computing. Without these feedbacks the configuration space of the neural network would not be rich enough for reservoir computing.
- If a dynamical system with a relatively simple configuration space is equipped with a delayed feedback, then its configuration space can become infinitely dimensional (e.g. see Section 9 in [21]). In general, dynamical systems with delayed feedback are still not well-understood, as argued in [22].
- Synchronization can be achieved through several forms of feedback (*cf.* [23] and references therein).

Thus in the SWEET setup the feedback mechanism is specifically exploited and further developed for information processing purposes in a rather systematic manner.

2.2. Possible advantages

The SWEET setup has the potential to be more powerful than the traditional sensing approaches. In particular, it might have the following advantages:

(1) *The sensing could be achieved even if the environment weakly influences the weaver.* While one naturally wishes to engineer a strong environment-sensor interaction, it might be possible to avoid this in the SWEET setup. Over time, the environment can leave a deep trace of its presence in the microscopic states of the sensing unit, the state weaver. Thus in the SWEET sensing approach the strength of the sensing unit - sensing target interaction is traded for possibly weaker but longer influence over time.
Above and throughout the text the terms 'weak' and 'strong' are used in a very precise sense: A judgement whether the interaction between two systems is 'weak' or 'strong' usually involves several criteria. One might try to identify a strong coupling constant that describes the dynamics of these systems, or inspect whether the changes in the macroscopic behavior occur when the systems are allowed to interact. However, the effect of the interaction between any two systems can be analyzed both at the microscopic and macroscopic levels. From the information processing point of view, the interaction does not have to 'show' macroscopically. It is sufficient that the changes in the structure of the super-reservoir states occur, i.e. that the entanglement exists at a microscopic level, and that these changes can be analyzed. It is possible that these microscopic changes become macroscopically visible over time, in some observable. The concept of the observable is often used in statistical physics to describe a property of the system that can be measured. In here, the term observable is used in the information processing context, to describe a measurable quantity relevant for information processing, e.g. as discussed in [6,24] (and references therein).

(2) *The interaction between the environment and the sensing unit does not have to be engineered in detail.* Since the sensing is done indirectly, the input signal can be chosen with a great amount of flexibility. This makes the process of carefully engineering the above mentioned interaction obsolete.

(3) *The environment might change as the result of the weaver-environment interaction.* This behavior could be seen both as a nuisance and a mechanism that could be exploited. In both cases the SWEET sensing approach might offer some distinctive advantages:

  (3a) Any sensor should influence the environment as little as possible since it is hard to measure a property of the system if the measuring apparatus strongly affects the system. *In the context of the SWEET sensing approach this requirement can be relaxed.* Once the input signal changes the state of the sensing reservoir, these changes could in turn influence the state of the environment, which might back-propagate to the sensing reservoir, by induces additional structural changes in its state. This set of weaver state changes could be erroneously interpreted as an alternation of the environment state. In the SWEET setup, such effects can be naturally dealt with by designing a special purpose delayed feedback to perform back-tracking and self-correction.

  (3b) *The weaver induced change of the environment could be used to realize 'plastic', or 'guided' sensing.* It is possible to use the sensor not only to analyze the environment, but also to change it, i.e. to pro-actively 'push' the environment to a desired state of interest, simultaneously. (This possibility is discussed in Section 3.5.4, where an example is also provided.)

To conclude, the benefits discussed above can have profound technological implications. The SWEET construct can free a sensing device engineer from practical constraints, and can provide much more flexibility regarding the choice of a suitable strategy for constructing new sensors. It could also enable novel reuses of the existing sensing solutions. The SWEET sensing setup simply ensures that every sensor is used at its maximum capacity.

3. Mathematical description of the SWEET algorithm

In this section the generic sensing concept is converted into a rigorous mathematical description of the sensing procedure. An algorithm is presented that provides an abstract description of the apparatus introduced in the previous section. The advantages of having an explicit algorithm are that the principles of device operation are made explicit, and separated from an implementation. The implementation can be achieved by analyzing the optimal way of engineering key mathematical abstractions. It is clear that the process of engineering will require solving a separate set of difficulties that are peculiar for every sensing situation. But with the algorithm at hand, at least there should be a clear set of engineering guidelines.

The algorithm is introduced in two steps. First, a generic formulation of the sensing problem is presented where time does not play a role. This is done just to provide a gentle introduction to a more complicated mathematical material that follows, where the concept of time is introduced as an essential ingredient, where the time sensitive features of the apparatus are used actively to infer about the state of the environment.

3.1. Generic mathematical definition of sensing

Mathematically, the most general way to define the sensing problem is as follows. For a given (microscopic) state of the environment q the goal is to characterize it in some way. This amounts to evaluating a function on the environment variable

$$\varphi = \phi(q) \tag{1}$$

where φ is the result of the characterization. For example, from positions of individual ions in a solution we should be able to compute the ion density and a mapping ϕ could be constructed so that $\varphi \approx 0$ describes a diluted system, while $\varphi \approx 1$ corresponds to a dense ionic solution. As this example illustrates it is often simple to construct the form of the mapping ϕ, but the problem is in providing the input: It is hard to measure the microscopic internal degrees of freedom of the environment q. This is the reason why, even if the form of ϕ were known, it would not be practical to use Equation (1) directly. Instead, the state of the environment needs to be inferred in some other way.

The traditional sensing apparatus consists of two components: an environment that one wishes to sense (the sensing target), and a dynamical system that reacts to the environment (the sensing unit). The state of the sensing unit depends on the state of the environment:

$$r = H(q) \tag{2}$$

The generic sensing approach works by monitoring the state of the sensing unit r. One cannot access q directly, but the strategy is to build an apparatus that can access some features of the environment by inspecting r. This mathematical formalization might appear artificial (e.g. the mapping ϕ was traded for another one, namely H), but it is extremely important for what follows, thus an example is in order to illustrate it. Though velocities of individual air molecules cannot be measured easily, it is still possible to build an equipment that will report on the pressure. In this example q would describe the velocities of air molecules, while r would correspond to the height of the mercury column exposed to the atmosphere.

The information about the state of the sensing unit, r, is further processed to infer the state of the environment

$$\varphi = \psi(r) \tag{3}$$

Taken together, mathematically this can be represented as a series of mappings, $q \to r \to \varphi$, that are carried out sequentially. Once engineered, these mappings should be performed after each other without time delays in between, otherwise loss in the sensing accuracy can be expected. Note that in

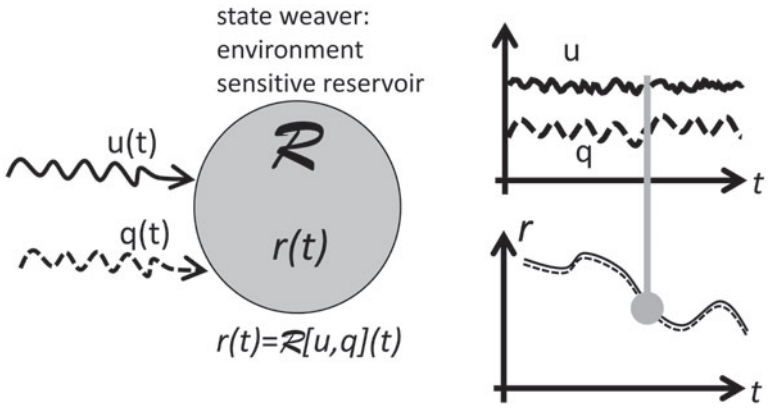


Figure 4. The environment sensitive reservoir acts as a filter. At every time instance t the value of the weaver configuration space coordinate r is determined by the form of the user provided input u, the delayed feedback f (not shown), and the environment variable q, prior to the time instance t.

this approach, the concept of time is not important for sensing *per se*, nor it is actively exploited: These mappings should be carried out together, as a group. If one wishes to do so they can be repeated at different time instances for repeated assessment of the environment. But this mechanism still does not exploit time in any way to perform sensing.

3.2. Exploiting time to improve sensing: the sensing reservoir

In contrast to the traditional setup, the SWEET idea exploits time, and the generic time-dependence of variables inherent in any sensing process (*cf.* Fig. 2). The SWEET setup exploits the fact that the sensing environment (target) can interact with the sensing unit over a longer time period and they can influence each other. This influence can be accumulated over time, and the main function of the state weaver is to 'weave' these joint states of the environment and the sensing unit together.

Since the goal is to sense time varying environment it is essential to describe the concept of time and how the whole system evolves in time. It will be assumed that the environment transits through a series of states q_t with times $t \in (-\infty, +\infty)$.

3.2.1. Important mathematical notation

It is easier to describe the algorithm using the discrete time representation, though this is not necessary. Thus without loss of generality it will be assumed that the environment changes at discrete time steps: the time variable t is only allowed discrete values $t \in \mathbf{Z}$ with $\mathbf{Z} = \{\cdots, -3, -2, -1, 0, 1, 2, 3, \cdots\}$.

In this discrete formulation all dynamical variables are treated as infinite sequences of values. An infinite sequence of values x_t with varying t will be denoted by

$$\mathfrak{x} \equiv \cdots, x_{t+2}, x_{t+1}, x_t, x_{t-1}, x_{t-2}, \cdots \tag{4}$$

and an ordered subset of such values in the range $a \leq t \leq b$ as

$$\mathfrak{x}_{b:a} \equiv x_b, x_{b-1}, x_{b-2}, \cdots, x_{a+2}, x_{a+1}, x_a \tag{5}$$

Such sequences will be used to describe how the states of the sensing apparatus components change in time. For future notational convenience the states will be always listed with more recent states first.

Symbols $\mathfrak{x}$ and $\mathfrak{x}_{b:a}$ will be referred to as a history of variable x. Occasionally when $\mathfrak{x}_{b:a}$ is used, it will be referred to as a finite, or semi-infinite history, depending on the choice of the indices. Clearly the most important sequence of interest is the one that describes how the environment evolves in time: $\mathfrak{q} \equiv \cdots, q_{t+2}, q_{t+1}, q_t, q_{t-1}, q_{t-2}, \cdots$.

3.2.2. The super-reservoir as a dynamical system

Influence of the environment on the weaver (E-on-W influence): Given a time instance t, what is the state of the system at this time instance, r_t, depending on? Since the sensing unit is assumed to be a dynamical system, the present state must depend on the past states: r_{t-1}, r_{t-2}, etc. A mathematically convenient way of presenting this is to assume that there is a mapping that can be used to convert an earlier state into the current one, e.g. like the Hamiltonian that describes the dynamics of a mechanical system. In the discrete case the rule that governs the dynamics of the weaver W state reads $r_{t+1} = H_W(r_t)$.

However, if the sensing unit (weaver W) and the environment E interact, then the state of the sensing unit cannot be just dependent on the present state of the environment, q_t, but it should be also dependent on the previous environment states: q_{t-1}, q_{t-2}, etc. This suggests that the proper equation to use is

$$r_{t+1} = H_W(r_t, q_t, u_t) \tag{6}$$

where u_t denotes the user provided input. Note that by using the equation above recursively, at each time instance t the state of the reservoir can be written as

$$r_t = H_{W,\tau}(q_{t-1:t-\tau}, u_{t-1:t-\tau}|r_{t-\tau}) \tag{7}$$

For example, with one iteration step ($\tau = 1$) one retrieves the definition of H_W and $H_{W,1}(q_{t-1:t-1}, u_{t-1:t-1}|r_{t-1}) = H_W(r_{t-1}, q_{t-1}, u_{t-1})$. After two iteration steps ($\tau = 2$) the iteration procedure results in $H_{W,2}(q_{t-1:t-2}, u_{t-1:t-2}|r_{t-2}) = H_W\left(H_W(r_{t-2}, q_{t-2}, u_{t-2}), q_{t-1}, u_{t-1}\right)$. In this way, the form of $H_{W,\tau}$ can be found for any $\tau \geq 1$. Note that it is tempting to drop the symbol $r_{t-\tau}$ in Equation (7), but this can only be done after the limit $\tau \to \infty$ has been taken, provided the system has the echo state property. (This will be discussed in Section 3.4).

Influence of the weaver on the environment (W-on-E influence): It is possible that the sensing unit also influences the environment. Then, Equation (6) needs to be augmented with

$$q_{t+1} = H_E(q_t, r_t) \tag{8}$$

Should this be the case the proper model has to include both equations. The SWEET sensing setup allows for this possibility. In the following, to simplify the discussions, it will be assumed that the weaver exerts a weak influence on the environment. Equation (8) will feature as an 'afterthought' to Equation (6).

3.2.3. The output of (sensing) computation

The variable r accumulates the information about the environment. This information can be further processed to characterize the environment, essentially in two ways, by either monitoring the present state of the sensing unit

$$y_t = \psi(r_t) \tag{9}$$

or by observing a series of such states of length $\tau + 1$

$$y_t = \psi_\tau(r_{t:t-\tau}) \tag{10}$$

The variable y_t denotes the response (output) of the device. The device is driven by a user defined signal u, which serves as the input to the computation performed by the SWEET sensor (the environment is also a source of input but this input cannot be controlled).

In the spirit of the reservoir computing literature the above inference procedures will be referred to as the readout. The main lesson from various applications of reservoir computing is that the implementation in Equation (10) is not necessarily better than the one given in Equation (9). An intuitive explanation is that r_t includes a 'memory' of the earlier states $r_{t-1}, r_{t-2}, \cdots$ anyway. Thus it is

reasonable to expect that implementation (9) can lead to very powerful sensors too, which from the engineering point of view might be easier to implement, e.g. for embedded applications. This is the form of the readout that will be assumed in the following.

3.3. The delayed feedback

The most natural way to incorporate the presence of the internal feedback $\mathcal{D}$ is to change Equation (6) into

$$r_{t+1} = H_W^{\mathcal{D}}(r_t, q_t, u_t; f_{t:t-\delta}) \tag{11}$$

where the symbol $\delta = 0, 1, 2, 3, \cdots$ specifies the degree of the delay. Note that $f_{t:t-\delta}$ represents a series of values, to be referred to as the feedback vector. Further, by assumption, only the readout layer can reach the internal states of the device, and this is an important design feature. The feedback layer should operate under the same principle, and be allowed to access the internal states only through a readout layer, possibly its own. Accordingly, one could design a special readout layer $\psi_{\mathcal{D}}$ that can access some features of the phase space that are relevant for feedback:

$$f_t = \psi_{\mathcal{D}}(r_t) \tag{12}$$

In the following all mappings that can be described by Equations (11) and (12) will be collectively referred to as the delayed feedback.

The above construct is flexible enough to cover a wide class of possible feedback types. For example, the mapping $H_W^{\mathcal{D}}$ could be constructed to take only the first or the last value in the feedback vector list $f_{t:t-\delta} \equiv (f_t, f_{t-1}, \cdots, f_{t-\delta})$; resulting in $H_W^{\mathcal{D}}(r_t, q_t, u_t; f_t)$ or $H_W^{\mathcal{D}}(r_t, q_t, u_t; f_{t-\delta})$ respectively. The first case represents a pure feedback (e.g. a one-step delay is usually not referred to as a delayed feedback). The second case is an instance of a genuine delayed feedback.

The delayed feedback operates in an autonomous manner. Once designed, it becomes an integral part of the device. It is rather challenging to argue about the optimal feedback choices in generic terms. This is something that will have to be left for future studies.

3.4. The filter concept

In what follows, it is useful to introduce the concept of the filter. For example, by using the filter notation, the SWEET sensing algorithm can be presented in a very concise way.

A filter $\mathcal{F}$ maps an infinite sequence of values $\mathfrak{u}$ into another sequence $\mathfrak{r}$, and the individual sequence values can be inspected as $r_t = \mathfrak{r}(t)$ and $u_t = \mathfrak{u}(t)$. For the future convenience the input and the output of the filter are denoted by the same symbols used to describe the dynamics of the super-reservoir (Section 3.2.2). It will be shown later that the super-reservoir realizes a filter. The following notation describes how a filter operates:

$$r_t \equiv \mathfrak{r}(t) = \mathcal{F}[\mathfrak{u}](t) \tag{13}$$

A filter process a sequence of values like a human would do. A human would inspect the sequence of values until a certain time, in a finite time window τ, and assign an output value depending on what has been seen. There is an important difference: a filter is a mathematical object assumed to have the capacity to 'remember' an infinitely distant past.

Any iterative map has the potential to realize a filter (converse might not be true). In particular, the equations that describe the dynamics of the super-reservoir represent a filter, but under specific conditions. To understand these conditions, assume that the filter $\mathcal{F}$ in Equation (13) is realized by the following map

$$r_{t+1} = F(r_t, u_t) \tag{14}$$

with $t \in \mathbf{Z}$. The question is, where should the sequence $\mathfrak{r}$ start, i.e what is the initial condition for the above map? The usual practice is to assume that the sequence starts from an infinitely distant past, i.e. that the limit $\hat{r} \equiv \lim_{t\to\infty} r_{-t}$ makes sense.

The mapping in Equation (14) can be iterated an arbitrary number of times (over index t), which leads to

$$r_t = F_\tau[u_{t-1:t-\tau}|r_{t-\tau}] \tag{15}$$

with $\tau \geq 1$. The function F_τ has τ arguments $(u_{t-1}, \cdots, u_{t-\tau})$ that are defined by the input, and one extra variable $r_{t-\tau}$ which describes the state (of the system the map describes) τ steps in the past. The filter $\mathcal{F}$ can be represented as the limit of F_τ when the number of arguments becomes infinite,

$$\mathcal{F}[\mathfrak{u}|\hat{r}](t) \equiv \lim_{\tau\to\infty} F_\tau[u_{t-1:t-\tau}|r_{t-\tau}] \tag{16}$$

Note the explicit dependence on the initial condition $\hat{r}$ in the infinite past on the left hand side of the equation.

When does the limit in Equation (16) exist? From a rigorous mathematical point of view this is a rather complicated question, since one needs to consider a sequence of functions with changing number of arguments, ending up with a non-compact domain (the space of infinite sequences is not compact). Nevertheless, Equation (16) is a useful pedagogical tool, and it will be used solely to illustrate an important principle. Assuming that the equation can be used in this informal way, the limit defines a filter if the initial condition variable $\hat{r}$ can be ignored. By definition, this is allowed when the system (the mapping in Equation 14) has the *echo state property*, and then the following replacement is valid

$$\mathcal{F}[\mathfrak{u}|\hat{r}] \to \mathcal{F}[\mathfrak{u}] \tag{17}$$

To summarize, when the replacement above can be safely made, then the mapping in Equation (14) indeed defines a filter.

Most dynamical systems in nature realize a filter, but one can easily construct counterexamples.[6] As an illustration, the mapping $F(r,u) = r$ produces no filter, since this mapping results in $\mathcal{F}[\mathfrak{u}|\hat{r}] = \hat{r}$ and the infinitely distant initial condition $\hat{r}$ cannot be ignored. As a positive example, the mapping $F(r,u) = u$ defines a filter. This filter has a very short memory, it only 'remembers' the last input: $\mathcal{F}[\mathfrak{u}](t) = \mathfrak{u}(t-1)$.

In the following it will be silently assumed that the causality principle holds: the output of the filter $\mathcal{F}$ at time t, given by $\mathcal{F}[\mathfrak{u}](t)$, cannot depend on the values of the input sequence at the same or later times: $t, t+1, t+2, \cdots$. Formally, if one replaces the values $\mathfrak{u}_{\infty:t}$ in the full history $\mathfrak{u}$ in some arbitrary way and calls the sequence obtained in this way $\mathfrak{u}'$, then it has to hold that $\mathcal{F}[\mathfrak{u}](t) = \mathcal{F}[\mathfrak{u}'](t)$. In the following it will be assumed that all filters have this property.

3.5. The sensing procedure description

It is important to realize that *the information one wishes to infer about the environment can be represented as a filter.* This filter is a user-defined object, and in the following it will be referred to as a *sensing goal*. The state weaving machine also acts as a filter. The key objective in the context of the SWEET sensing setup is to adjust the weaving machine (filter) so that it can represent the sensing goal (filter).

3.5.1. The sensing goal as a filter

Any sensing procedure is equivalent to the information processing task where the goal is to compute some feature function on the history of the environment microstates up to an arbitrary time instance t. In the on-line sensing mode, the goal is to do this at every time instance t. In mathematical terms, this means that one wishes to evaluate the following expression:

$$\varphi_t = \phi[\mathfrak{q}](t) \tag{18}$$

where the filter ϕ acts on the sequence of the states of the environment, and *is defined by the observer* who studies the system. The collection of such filters will be denoted by Φ. As an example of an element in this collection, consider a filter that outputs 1 if the time series it observes up to a certain time point is periodic, and returns 0 otherwise.

3.5.2. The sensing reservoir as a filter

The state weaver also acts as a filter, as illustrated in Fig. 4. At every time instance t the value of the reservoir state r is determined by the form of the input u and the environment q up to that point in time. Using the filter notation introduced earlier, one can write

$$r_t = \mathcal{R}[\mathrm{u}, \mathrm{q}](t) \tag{19}$$

where $\mathcal{R}$ denotes the filter object, a mathematical representation of the reservoir.

Equation (19) can be obtained by following the procedure described in Section 3.4. First, the limit $\tau \to \infty$ is taken of Equation (7), and then the echo state property is assumed. (As discussed earlier, without assuming the echo state property of the reservoir one would have to use another form, $r_t = \mathcal{R}[\mathrm{u}, \mathrm{q}|\hat{r}](t)$, where $\hat{r}$ denotes the initial state of the reservoir in the infinitely distant past.)

Occasionally, to emphasize that a reservoir $\mathcal{R}$ is equipped with a specific delayed feedback mechanism, $\mathcal{D}$, the symbol $\mathcal{R}_\mathcal{D}$ will be used.

3.5.3. The SWEET algorithm as a filter learning procedure

Now the final part of the algorithm can be stated. The algorithm is designed to systematically address the following difficulties: (1) The states of the environment cannot be directly assessed, and in practice one can never use Equation (18) directly since q (note the history notation) is not known. (2) The information about q is stored (encoded) in r_t (note the absence of the history notation). The problem is that the readout layer might not be able to assess all intrinsic features or r_t, as not everything can be measured about the system. How can one extract (decode) the information about the environment then? The answer is simple, and yet profound as will be discussed later when the expressive power of the setup will be analyzed: *In principle, it should be possible to exploit the freedom in choosing the drive u and, under some constraints on the reservoir, realize any sensing goal.* (These constraints are discussed later in Section 4.2.)

If one can find the drive that rearranges the internal states of the reservoir, the readout layer could extract the information about the environment. The SWEET sensing algorithm is built on this insight, and this is precisely what is being illustrated in Fig. 3. In mathematical terms the SWEET sensor implements a particular inference mapping

$$y_t = \psi(\mathcal{R}_\mathcal{D}[\mathrm{u}, \mathrm{q}](t)) \tag{20}$$

and, in fact, a series of mappings that can be generated by all possible choices of the signal weaver $\mathcal{R}$, the drive signal u, the feedback layer $\mathcal{D}$, and the readout layer ψ. Can one choose these so that $y_t = \varphi_t$? In the context of this question it is important to consider the designs with a relatively simple feedback layer, otherwise the feedback layer might do all the computation. (By construction, the readout layer does not operate as a filter, and cannot perform any substantial computation in that sense.)

In practice, it would be hard to choose the reservoir $\mathcal{R}$ freely, as it is most likely given by the engineering aspects of a sensing problem at hand. The only real freedom one has, is to choose a particular input signal u_* so that

$$\phi[\mathrm{q}](t) \approx \psi(\mathcal{R}[\mathrm{u}_*, \mathrm{q}](t)) \tag{21}$$

The above equation represents the most abstract formulation of the SWEET sensing algorithm. As advocated earlier, the user provided input signal does three operations at the same time: decoding, extraction, and analysis. This, naturally, raises a question: Can the equation above be turned into an equality? which will be addressed in Section 4.

3.5.4. The plastic sensing

The roles of the environment and the reservoir are not necessarily symmetric. While small changes in the environment might induce large changes in the reservoir, the converse does not need to hold. In situations where the weaver is much smaller than the environment, one can expect a separation of scales in the sense that it takes less time for the environment to influence the weaver than the other way around. For example, if the environment one wishes to sense is a solution containing ions, it would be hard to change the ion flow if a small object (e.g. an electrode controlled by an input signal) is inserted into the solution. In some sense, the environment is more 'inert' than the weaver, and yet it could be influenced by it. In this way, Equation (8) appears as an 'afterthought' to Equation (19).

Thus one can envision situations where during a limited time interval the environment can be assumed constant but it can still change on longer time scales (the adiabatic approximation). This could be described by the following model (the plastic sensing principle)

$$r_t = \mathcal{R}[\mathfrak{u}, \mathfrak{q}](t) \tag{22}$$

$$y_t = \psi(r_t) \tag{23}$$

$$q_{t+1} = q_t + \epsilon \hat{H}_E(q_t, r_t) \tag{24}$$

The first two equations describe the sensing part of the algorithm. The last Equation (24) describes how the environmental condition changes (which is just a special form of Equation 8), where ϵ is assumed small in some sense. Interestingly, this implies that the sensor while 'analyzing' might also gradually 'encourage' the environment to be characterized in a desired way.

As an illustration, consider the following example. Imagine a sensor designed to analyze ionic solution that operates by measuring the impendence spectra between two electrodes that have been inserted in the ionic solution. An external alternating voltage used to obtain the spectra could also induce changes in the ion concentration profile. These would be likely highly fluctuating, but it is possible that over time a distinct tendency in the ion concentration profile develops. Thus the input signal used for sensing also changes the environment.

Note that if the drive is too strong then the state of the environment becomes completely defined by the drive. This implies that in the expression

$$\mathfrak{q}(t) = \mathcal{Q}[\mathfrak{u}|\hat{q}](t) \tag{25}$$

the dependence on the initial condition of the environment $\hat{q}$ is completely lost leading to $\mathcal{Q}[\mathfrak{u}|\hat{q}] = \mathcal{Q}[\mathfrak{u}]$, resulting in a strong echo state property which is ultimately undesirable since it renders the sensor useless. Of course, this is not a problem if one is not interested in sensing, and only wishes to use the sensor to modify the environment.

4. The expressive power of the SWEET sensing model

The choice of the drive is strongly related to the expressive power of the sensing setup. For example, if a complex reservoir is used, with a rich set of states, it is likely that relatively simple drive signals will do. On the other hand, for simple reservoirs, more complicated drive signals might be required to achieve the same quality of sensing. A natural follow-up question is whether for some combinations of the environment and the sensing reservoir it is not possible to find a useful drive. It is virtually impossible to discuss the expressive power of a model of computation without using rigorous mathematics, and the SWEET sensing model is no exception. Nevertheless, an attempt will be made to discuss the expressive power of the SWEET setup in an intuitive manner while maintaining a clear connection with the underlying mathematical concepts.

The usual strategy of arguing for a limitless expressive power is to exploit the Stone-Weierstrass approximation theorem, and show that the model of computation one wishes to analyze complies

with the assumptions of the theorem. The same will be done in here, but instead of focusing on mathematical concepts *per se*, these will be discussed from an engineering point of view. In particular, the engineering requirements needed to realize the assumptions of the theorem will be analyzed. This should give a rough idea regarding where the practical boundaries of the SWEET sensing approach are.

4.1. Engineering mathematical assumptions

To analyze the expressive power of the SWEET sensing setup a useful form of the the Stone-Weierstrass approximation theorem will be stated first.

Theorem (Stone-Weierstrass): Let $\mathcal{W}$ denote an algebra of functions that map from a domain $\mathcal{Q}$ to real numbers; $\mathcal{W} = \{a | a : \mathcal{Q} \to \mathbf{R}\}$. If the algebra $\mathcal{W}$ is continuous, the domain $\mathcal{Q}$ it operates on compact, contains a constant element, and separates points, it can approximate any function $\phi : \mathcal{Q} \to \mathbf{R}$ uniformly: For any accuracy requirement $\epsilon > 0$ and a target function ϕ one can find an element in the algebra a_* so that $|a_*(\mathfrak{q}) - \phi(\mathfrak{q})| < \epsilon$ for every $\mathfrak{q} \in \mathcal{Q}$.

Since the compactness of the domain $\mathcal{Q}$ is crucial it will be assumed, as usual in the reservoir computing literature, that only filters with the fading memory property are of interest. This assumption is necessary since the space of infinite (real valued) sequences is not compact. Thus we assume that both the sensing devices and sensing goals have the fading memory property. This more or less takes care of the compactness requirement. Other requirements are discussed below.

Collection of weavers: Each sensing device, a state weaver, behaves as a filter, precisely in the way discussed earlier. Let $\mathcal{W}$ denote a collection of such filters. Every element a in the collection operates on the space $\mathcal{Q}$ of all possible environment histories $\mathfrak{q}$. Each element is defined by the choice of the sensing reservoir, the delayed feedback, the readout layer, and the input drive:

$$a \equiv (\mathcal{R}_a, \mathcal{D}_a, \psi_a, u_a) \tag{26}$$

An element a acts on environment histories as follows

$$a[\mathfrak{q}](t) = \psi_a(\mathcal{R}_a[\mathfrak{u}_a, \mathfrak{q}](t)),\ \mathfrak{q} \in \mathcal{Q} \tag{27}$$

Sometimes, if the environment variable is of no interest for a discussion, to make the notation more compact, $a(t)$ will be written instead of $a[\mathfrak{q}](t)$.

What can be engineered in a particular sensing context defines the collection. It is reasonable to expect that one has limitless resources to realize readout layers and drives. There are two scenarios related to the use of the reservoir:

Scenario 1: It is possible to build a set of sensing reservoirs.

Scenario 2: One is forced to work with only one sensing reservoir.

The second alternative is more restrictive. Likewise, there is an option to use one delayed feedback or one could try to construct many to spawn variability in the algebra. While clearly an option, in the context of the second scenario it is natural to assume that only one delayed feedback is available. One should restrict the computing power of such a feedback in some way. The main reason behind these assumptions is that one has to be careful to avoid the situation where the computation is done completely by the feedback layer.

Algebra of weavers: To form such an algebra, one should be able to build larger sensors from smaller ones, in a way that mimics the mathematical operations of addition, multiplication, and multiplication with a constant. The largest difference from the usual reservoir computing way of thinking is that, in here, every element is associated with a drive signal. So when two elements a and b are combined one must make a decision on, among many other things, how to combine their respective drive signals u_a and u_b. To illustrate the point, consider the situation where the goal is to engineer a binary algebraic operation on the sensor outputs, e.g. a multiplication $a(t) \times b(t)$ or an addition operation $a(t) + b(t)$.

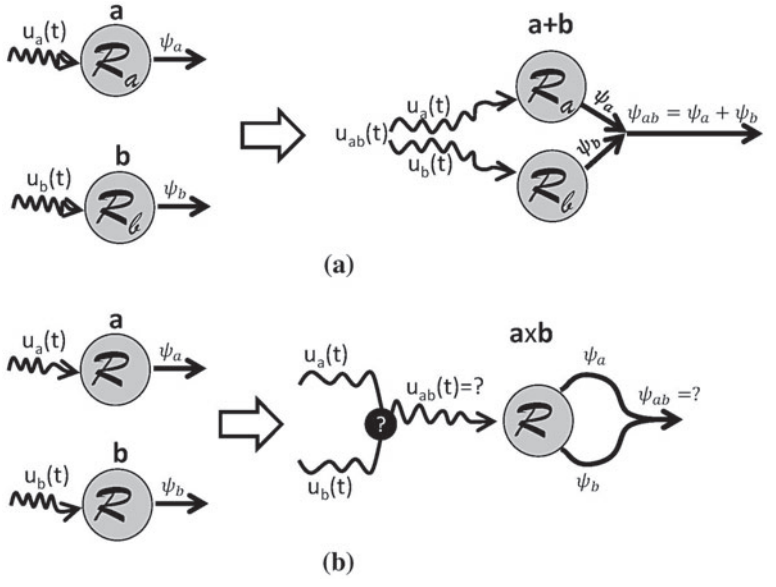


Figure 5. An illustration of the binary algebraic operations of addition (panel a) and multiplication (panel b) on sensors. Panel (a): With two distinct reservoirs it is much easier to identify the joint input and the readout layer. Panel (b): If only one reservoir is available, it is not clear what to choose as an input and a readout layer.

Consider the following example, in the context of *scenario 1*. The goal is to build from two sensors $a \equiv (\mathcal{R}_a, \mathcal{D}_a, u_a, \psi_a)$ and $b \equiv (\mathcal{R}_b, \mathcal{D}_b, u_b, \psi_b)$, a third (replacement) sensor that will be denoted by $a \circ b \equiv (\mathcal{R}_{ab}, \mathcal{D}_{ab}, u_{ab}, \psi_{ab})$ that outputs $(a \circ b)(t) \equiv a(t) + b(t)$. To build such a sensor one simply forms a tensor product of the inputs $u_{ab}(t) \equiv (u_a(t), u_b(t))$, and lets the readout layers achieve the desired algebraic operations, as illustrated in Fig. 5, panel (a). However, it is much harder to engineer such a process when the number of devices one wishes to combine increases. The concept does not scale well. This wiring-like problem might be an issue in embedded applications, but there are no conceptual barriers in realising this scenario.

In contrast with the first scenario, the presence of a drive signal attached to every element of the algebra creates a range of problems in the context of *scenario 2*. In particular, tracing carefully how inputs should be combined is much harder.

Consider the following example, illustrated in Fig. 5, panel (b). Assume that the goal is to build from two sensors $a \equiv (\mathcal{R}, \mathcal{D}, u_a, \psi_a)$ and $b \equiv (\mathcal{R}, \mathcal{D}, u_b, \psi_b)$, a third (replacement) sensor that will be denoted by $a \circ b \equiv (\mathcal{R}, \mathcal{D}, u_{ab}, \psi_{ab})$ that outputs the product of the respective outputs: $(a \circ b)(t) \equiv a(t) \times b(t)$. Note that, as advocated earlier, the same delayed feedback is used to define the elements a and b. The version where the delayed feedback is fixed is more useful in situations where embedded applications are of interest. While it is clear that the operation on the readout layers should resemble the binary operation on real numbers, i.e. $\psi_{ab} = \psi_a \times \psi_b$, it is much less clear what should u_{ab} correspond to: there is no reason to believe that $u_{ab}(t) = u_a(t) \times u_b(t)$ will be a suitable drive.

Separation property: Out of all properties of the algebra that need to be assumed, the most interesting one is the separation property. For any two arbitrary environmental conditions $\mathfrak{q}_1$ and $\mathfrak{q}_2$ that are different, $\mathfrak{q}_1 \neq \mathfrak{q}_2$, there has to be an element $a_{12} \in \mathcal{W}$ such that $a_{12}[\mathfrak{q}_1] \neq a_{12}[\mathfrak{q}_2]$. For *scenario 1* this is not such a strong requirement. If sensing reservoirs can be chosen at will this property should be easy to realize. In the case of *scenario 2*, the separation property needs to be achieved by finding a drive $\mathfrak{u}_{12}$ such that $\psi(\mathcal{R}[\mathfrak{u}_{12}, \mathfrak{q}_1](t)) \neq \psi(\mathcal{R}[\mathfrak{u}_{12}, \mathfrak{q}_2](t))$ for some t values. This principle is exactly the one illustrated in Fig. 3.

4.2. The expressive power theorems

The above discussions can be summarized as a theorem and a conjecture. The theorem deals with *scenario 1* and is trivial to prove. The conditions on the state weavers are rather generic. The conjecture deals with *scenario 2* but, to prove it, less generic state weavers need to be assumed.

Theorem (*scenario 1*): Let $\mathcal{W}$ be algebra of weavers that separates points, and where each element has the fading memory property, and is continuous. Let Φ be a collection of all possible sensing goals (again, continuous, and with the fading memory property). Then for every sensing goal $\phi \in \Phi$, and an accuracy requirement $\epsilon > 0$, it is possible to find a weaver $a \in \mathcal{W}$, such that

$$|a[\mathrm{q}](t) - \phi[\mathrm{q}](t)| < \epsilon \tag{28}$$

for every environment conditions q and every time t.

The proof is trivial. Let the algebra be constructed from weavers with a constant drive. Then exactly the same proof strategy as for the LSM model can be used. In fact, the constant drive assumption cannot be avoided. If not assumed, it emerges naturally while implementing the proof strategy.

Given one can choose among many weavers, the suggested sensing paradigm has infinite sensing power. However, the constant drive assumption contradicts the SWEET sensing principle. Essentially, the expressive power comes from the ability to build sensing reservoirs at will. Thus the question is whether the same expressive power can be archived when the choice of sensing reservoirs is limited, e.g. when only one can be chosen. The following conjecture describes that situation.

Conjecture (*scenario 2*): Assume that a sufficiently complex SWEET sensor is given, in the usual reservoir computing sense.[6] The algebraic operations of the algebra are achieved by combining readout layers, drives, and feedback mechanisms. Let the algebra separate points. Then for any given characterization function ϕ it is possible to find a choice of the readout layer ψ, the drive signal u, and the delayed feedback mechanism $\mathcal{D}$, such that

$$|\psi(\mathcal{R}_{\mathcal{D}}[\mathrm{u}, \mathrm{q}](t)) - \Phi[\mathrm{q}](t)| < \epsilon \tag{29}$$

holds for any environment history q and time t.

The conjecture can be proven using the proof strategy detailed in [6]. However, a preliminary investigation shows that the conditions on the sensing reservoir are much more restrictive than in the standard reservoir computing setup. Stating these conditions in detail would require a rather space consuming formal and technical treatment which will be left for a forthcoming publication. In particular, the conditions on the class of delayed feedback layers assumed in the conjecture need to be carefully analyzed. For example, one can show that a special form of time-invariance needs to hold that maintains a synchronization between the drive signal and the environment. This synchronization could be achieved by designing a special purpose delayed feedback. In fact, in several numerical tests where memristor networks are used to realize the state weaver, it has been confirmed that the loss of the environment-input synchronization leads to the loss of the functionality of the sensor, which will be reported elsewhere.

5. Engineering the SWEET sensing algorithm

Perhaps the simplest procedure of engineering a SWEET sensor is to directly translate the abstract mathematical objects that constitute the SWEET sensing algorithm into device components. The sensor can be designed in three major steps:

(1) Choose an appropriate dynamical system that will work as the state weaver. This choice defines the mathematical form for the mappings (6) or (11).

(2) Engineer the readout strategy procedure. This leads to an engineering implementation of the equation (9). The readout layer can be constructed *in silico*, or some hardware implementation could be used.
(3) Consider possible feedback strategies, i.e envision suitable implementations of Equations (11) and (12), and the useful drive (response initiator) signals.

By far the most challenging step is the last one, which we proceed to discuss in a bit more depth. As stated earlier, the delayed feedback and the drive signal are strongly related and it is hard to discuss how to find one of them without the other.

5.1. Finding optimal drives (user provided and the delayed feedback)

The key component of the SWEET sensing setup is the idea of the delayed feedback and the external drive that realizes indirect sensing. The question is whether it is possible to envision a generic procedure for finding these. It is likely that the choice of these will be have to be addressed separately for each sensing scenario. But one should be able to answer this question at least in principle. There are some ideas from reservoir computing that can be used as guidelines for doing that.

It might not be obvious that the choice of the suitable drive is strongly related to the choice of the feedback. It is perfectly plausible that in some applications they might be viewed as one and the same. This can be illustrated by envisioning a range of possibilities in which the user provided input and the delayed feedback change roles easily. For example, the obvious trivial case is when the delayed feedback is absent. The middle ground corresponds to a situation where the drive signal $\mathfrak{u}$ drives the device, and the automated delayed feedback just corrects it slightly, i.e to achieve online computation. As yet another extreme one might consider a situation where the drive signal $\mathfrak{u}$ is completely absorbed into the delayed feedback. The delayed feedback mapping, Equations (11) and (12), can be designed to automatically produce an approximation to $\mathfrak{u}$.

One should critically reflect on whether the feedback is doing all the sensing and not the sensing reservoir, e.g. by comparing the computational complexity of the delayed feedback and the sensing reservoir components. These considerations might be extremely important in the context of embedded sensing. Thus an embedded sensing application will likely require other forms of feedback than other non-embedded versions of sensing.

A straight forward strategy of choosing useful feedback mechanisms is to specifically pick the ones that can increase the quality of the sensing reservoir. The issue of the reservoir quality has been addressed in the literature, and there are several criteria that could be targeted. However, the most important gross feature that makes a good reservoir is the non-linearity of the dynamics. This is exactly the mechanism that the Echo State Network paradigm exploits. It is a well-known insight that one does not need to train the full recurrent neural network, just its readout layer, provided the network is complex enough. Clearly the 'intelligence' that performs computation is stored in intricate feedback loops of such networks. There is clearly an option of adding such loops explicitly if they are not present. The SWEET sensing setup exploits this fact in a rather systematic manner.

5.2. Generic modes of using the sensor

The expressive power of the SWEET sensing setup has been investigated starting from the mathematical side of the problem, which was followed by the analysis of the related engineering issues. Here, the goal is to start from the entirely opposite point of view, and start from concrete practical problems. There are two ways to exploit the SWEET sensing setup, and to describe them some terminology from the machine learning field will be used: supervised versus unsupervised learning strategy.

In the supervised mode the user tests the device against known environmental conditions. A typical example would be a classification problem, where one wishes to classify an unknown environment in terms of a finite number of classes. To do this, one trains the sensor on a known class of environmental conditions, and use it later to generalize on an unknown environment.

The key point is that in a supervised learning setup one is given an opportunity to search for the right input signals together with a suitable delayed feedback by observing the output of the sensor under the known environmental conditions. The training procedure consists of the following: The input signal is altered until the agreement is found with the desired output of the computation (sensing). Clearly, having a theory of device operation is invaluable, as this can be done through simulation, on the computer, by re-using the numerous techniques available in the machine learning literature, but with a limitation that one should exploit the ones that deal with time series data analysis. It would be hard to provide a generic discussion how this can be done, but as an illustration, consider the following.

The simplest technique to find the signals and the feedback mechanism is to use the exhaustive random search: one simply tries signals at random until an agreement is found. It is further possible to augment such a random search procedure with a guidance by exploiting genetic algorithms. For example, in the context of SWEET sensing setup, the genetic algorithm approach can be implemented in a rather straight forward manner: one simply needs to realize the usual mutation and crossing operations on the space of drive signals and feedback mechanisms.

In the unsupervised learning mode, the user does not have any information about the environment. Instead, the goal is to infer a probability distribution of the environmental signals. Again, there are several techniques from machine learning that might be re-used in the SWEET sensing context that would have to be adapted to the time series data analysis. There are many possibilities.

Example 1: As an illustration consider the following. One could use an artificial neural network to realize the delayed feedback by using the Hebbian learning rule. All the input to the device is provided by the feedback signal. The external drive signal is absent. Any other system could be used instead of the neural network, provided it exhibits a form of synaptic plasticity (e.g. allows for an implementation of the Hebbian learning). Then, the trends in the dynamics that are persistent will be emphasized, first in the delayed feedback object, and then self consistently with the environment.

Example 2: In connection with Example 1, one could also try a hybrid method, a combination of supervised and unsupervised learning. In particular, the recent success of deep learning techniques illustrated that such a combination can be very powerful. In the context of the SWEET sensing setup this would imply that the unsupervised learning step would be used to extract classes of typical environmental signals. Then, in the final supervised learning step, these classes would be labeled.

6. Discussion

A novel concept of sensing has been introduced, the SWEET sensing setup, and described both at the intuitive level, and in precise mathematical terms. Some key features of the setup are highlighted below.

Owing to the lack of demands on the specific sensor-environment interaction, the SWEET sensing approach is flexible and could be used in a range of sensing applications where an environment of interest needs to be analyzed over time. Intuitively, the key idea that the SWEET sensing setup is built on is to trade a weak instantaneous interaction between the sensor and the environment, with a prolonged, possibly much weaker, interaction over time. The ideas presented in Fig. 2 can be used as guidelines to build an actual device. The SWEET sensing algorithm (Equation 21 and the related constructs) has been proposed as a way of representing the intuitive concepts illustrated in the figure as well-defied mathematical objects. Then, these objects can be engineered in the context of each specific sensing application. Ultimately, the algorithm can be viewed as a 'user manual' for implementing the SWEET sensing idea. Situations in which the SWEET sensing approach could be used with an advantage (over the traditional sensing setup) might include sensing problems where engineering suitable environment-sensor interactions is problematic due to technological limitations (e.g. as in embedded applications), or when the sensor is made of dynamical components with many degrees of freedom, with weak, unknown, or random interactions.

As a concept, indirect sensing is not widely used. Most applications exploit the liner response theory, which are indeed numerous. It is somewhat surprising that the possibilities of going beyond the linear response theory suggested in here have not been extensively investigated. An interesting application of indirect sensing has been suggested in [17]. The sensing procedure works by monitoring the changes in the structure of the feedback apparatus that controls the robot (e.g. by tracing over time the changes in motor torques, acceleration of the robot body, pressure sensor values).

The SWEET sensing setup features a specific implementation of indirect sensing, in the form of a generic extension of the linear response theory to time series data analysis. The SWEET equation (Equation 21, and the related mathematical expressions) is a concise description of this generalization. The question is whether other implementations are possible. It will be argued that to implement an indirect sensing approach one is more or less forced to adopt the SWEET equation. In some sense, the generic sensing hardware setup in Fig. 2, naturally encourages the mode of operation represented by the SWEET equation.

As an illustration of the above claims, assume that the goal is to make the indirect sensing concept of the SWEET equation as explicit and as generic as possible: An input u is provided to the system, and the response of the system depends on the state of the system r_t at a particular time instance. By assumption, there is a way to engineer a mapping that to every input u assigns an output y depending of the state of the sensing reservoir:

$$y = G(r_t, u) \tag{30}$$

The symbol G describes a 'gate' like function which is not necessarily linear. Such non-linear gates are frequently used in molecular electronics (e.g. the Nanocell construct suggested in [25] is a typical example), and have attracted a considerable attention as programmable gates. Interestingly, despite a possibility to use them for sensing, such non-linear gates have not been extensively studied for that purpose in the literature.

In the simplest setup, the non-linear gate construct could be used for sensing as follows. A natural procedure is to perform a series of measurements, where a user defined input, a real number $u \in \mathbf{R}$, is provided and the output is recorded, also a real number $y \in \mathbf{R}$. By analyzing a large number of (u, y) pairs obtained this way, one should be able to infer about the state of the reservoir r_t (and subsequently use this information to assess the state of the environment q_t). Exactly the same ideas are used in the linear response theory where the output is related to the input through a response function χ (instead of the non-linear function G).

In continuous time representation, and assuming time invariance, the fundamental equation is $y(t) = \int_{-\infty}^{t} \chi(t - t')u(t')dt'$. What makes the linear response theory interesting is that the response function $\chi(t)$ can be used to gain information about the system owing to the known connection between the response function χ and some properties of the system in equilibrium (the Kubo formula). [26] By having a model for χ and by measuring it, one infers the properties of the system. Here, the goal is to generalize this idea of indirect measurement in the context of the time series data analysis.

Ideally, one would like to measure as many (u, y) pairs as possible for a fixed t. This is the most generic indirect sensing setup. Though being perfectly feasible, there are two issues with this setup:

- It might be very hard to engineer a reservoir that can be driven this way, without altering its state r_t. Thus, probably, the idea of repeatedly performing (u, y) measurements at a fixed t has to be given up. In the SWEET setup, this is acknowledged by assuming that the drive indeed changes the state of the system (*cf.* Equation 19).
- The responsiveness of the gate might be a problem. For example, the output could be extremely sensitive to input for some values of r_t but it could also be insensitive for other values, and the efficiency of the analysis might depend strongly on the part of the configuration space the system occupies. In the SWEET setup these issues are circumvented by weaving the states of the environment and the sensing reservoir over time. Essentially, the SWEET algorithm is built by generalizing Equation (30) to account for state histories, i.e by emphasizing the filter construct.

While it is known from the literature that a delayed feedback leads to a more complex dynamical behavior, the idea to carefully engineer a (delayed) feedback mechanism for information processing purposes seems to be unexplored, despite several pioneering efforts reported in the literature. For example, in the context of reservoir computing the idea has been explored in a series of papers [8,19, 27–29] where the delayed feedback was used to increase the complexity of the reservoir. In [30], and a subsequent series of publications, the idea of using time-varying feedback was investigated in the context of neuromorphic computing. In the SWEET sensing setup, the delayed feedback mechanism is used to achieve specific functionalities: to synchronize the drive $\mathfrak{u}$ and the environment $\mathfrak{q}$ signals, and to ensure stronger weaving of states. It has been argued that the drive signal $\mathfrak{u}$, is just another face of the automatic delayed feedback mechanism, but they do play distinct roles.

The SWEET sensor can operate in two major modes: supervised and unsupervised. The first mode is useful in situations where it is possible to 'train' the sensor against a set of known environmental conditions, where one simply wishes sensor to generalize: to label an unknown environmental condition in a meaningful way. For example, a goal could be to identify deviations from the stationary state (two classes: 'in equilibrium', or 'out of equilibrium'). An unsupervised mode can be used to extract features from an unknown (stationary) environment. For example, if one wishes to extract typical environmental conditions.

Owing to the possibility of choosing the drive signal freely, a SWEET sensor could be adjusted ('programmed') for different sensing goals. In particular, if the state weaver can influence the environment an additional type of sensing applications can envisioned. A form of 'plastic sensing' or 'guided sensing' could be performed. While the drive signal is being learned the environment could slowly change. Thus in this setup the drive is used to achieve two things: to 'force' the environment to 'reveal' itself and, at the same time, 'guide' the environment to a certain state.

The SWEET sensing approach is generic and it could be applied to a variety of situations. For example, the related algorithm has been represented in the form of an *algorithmic template* in the sense that each instance of the algorithm exploitation (either for theoretical or experimental work) will require an implementation of some 'free elements'. These application dependent elements of the algorithm are the functions that describe various aspects of the dynamics, e.g. H_W, H_E and would have to be engineered for a specific sensing problem, or assumed if theoretical modeling is done (e.g. to aid in optimizing the sensor performance).

It is perfectly possible to adopt a pragmatic point of view and use the SWEET setup without worrying about its expressive power. A careful engineering of the equipment in each specific sensing situation usually leaves a lot of maneuvering space to achieve greater sensing power. However, an attempt has been made to address the following principal question(s): What are the limits of the SWEET sensing approach? Is it possible to identify classes of sensing applications that cannot be solved this way? This was done using intuitive reasoning as much as possible, while sill maintaining a clear connection with an underlying mathematical rigor that provided the background for the discussion.

It has been argued that the sensing paradigm has an infinite expressive power which, however, comes at a price: When compared to the classical reservoir computing setup, the conditions on the state weaver (the sensing reservoir) are more restrictive in the sense that in addition to the reasonably generic echo state and separation properties, one needs to assume that there is a mechanism to maintain the synchronization between the environmental and input signals, which can be realized by using the delayed feedback. The question is whether the unlimited expressive power is possible if a computationally limited class of delayed feedbacks is assumed. This will be investigated in forthcoming publications.

Note

1. The super-reservoir concept was conceived in the period 2013–2014 during the proposal preparation process for the RECORD-IT project, which exploits and further develops the SWEET sensing algorithm in the context of ion detection. The goal is to build a proof of the concept device, essentially an ion-sensitive reservoir, and exploit

the SWEET sensing algorithm to characterize the distribution of ions in various physiological solutions. One can envision numerous technological applications of the RECORD-IT apparatus in a range of areas such as medicine or biotechnology.

Acknowledgements

The author wishes to acknowledge a series of discussions with all partners in the RECORD-IT consortium that helped to formulate the initial ideas. These were further refined through subsequent discussions. The process helped in shaping the robust mathematical background that is being presented in here.

Funding

This work was supported by an FET Open HORIZON 2020 grant RECORD-IT (Reservoir Computing with Real-time Data for future IT) [grant number 664786].

ORCID


Zoran Konkoli http://orcid.org/0000-0002-5468-1388

References

[1] Campenhout JV, Schrauwen B, Verstraeten D. An overview of reservoir computing: theory, applications and implementations. In: 15th European Symposium on Artificial Neural Networks (ESANN2007-15). p. ES2007-8. Available from: http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2007-8.pdf
[2] Kulkarni MS, Teuscher C. Memristor-based reservoir computing. In: 2012 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), Amsterdam. p. 226–232.
[3] Jaeger H, Lukoševicius M, Schrauwen B. Reservoir computing trends. KI. 2012;26:365–371.
[4] Lukoševicius M, Jaeger H. Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. 2009;3:127–149.
[5] ORGANIC-EU-FP7. Reservoir computing: shaping dynamics into information, 2009. Available from: http://reservoir-computing.org.
[6] Konkoli Z. On reservoir computing: from mathematical foundations to unconventional applications. In: Adamatzky A, editor. Advances in unconventional computating. Vol. 1, Theory. Springer; 2016.
[7] Li T, Nakajima K, Cianchetti M, et al. Behavior switching using reservoir computing for a soft robotic arm. In: 2012 IEEE International Conference on Robotics and Automation (ICRA); 2012. p. 4918–4924.
[8] Hauser H, Ijspeert AJ, Fuchslin RM, et al. The role of feedback in morphological computation with compliant bodies. Biol. Cybern. 2012;106:595–613.
[9] Caluwaerts K, D'Haene M, Verstraeten D, et al. Locomotion without a brain: Physical reservoir computing in tensegrity structures. Artif. Life. 2012;19:35–66.
[10] Palumbo F, Barsocchi P, Gallicchio C, et al. Multisensor data fusion for activity recognition based on reservoir computing. In: Bota JA, Garcalvarez JA, Fujinami K, Barsocchi P, Riedel T, editors. Evaluating AAL systems through competitive benchmarking: international competitions and final workshop, EvAAL 2013, July and September 2013. Proceedings. Berlin: Springer, Berlin Heidelberg; 2013. p. 24–35.
[11] Bacciu D, Barsocchi P, Chessa S, et al. An experimental characterization of reservoir computing in ambient assisted living applications. Neural Comput. Appl. 2014;24:1451–1464.
[12] Sheik S, Marco S, Huerta R, et al. Continuous prediction in chemoresistive gas sensors using reservoir computing. In: EUROSENSORS 2014, the 28th European Conference on Solid-state Transducers; Vol. 87; Procedia Engineering, Brescia. p. 843–846.
[13] Hauser H, Fchslin RM, Nakajima K. The body as a computational resource. In: Hauser H, Rudolf MF, Rolf P, editors. Morphological computation. Self-published; 2014. p. 226–244.
[14] Fonollosa J, Sheik S, Huerta R, et al . Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. Sens. Actuators B-Chem. 2015;215:618–629.
[15] Mesaritakis C, Kapsalis A, Syvridis D. All-optical reservoir computing system based on ingaasp ring resonators for high-speed identification and optical routing in optical networks. In: Razeghi M, Tournie E, Brown GJ, editors. Quantum sensing and nanophotonic devices XII. Vol. 9370, Proceedings of SPIE. Bellingham: SPIE-International Society for Optical Engineering; 2015:937033.
[16] Wootton AJ, Day CR, Haycock PW. An echo state network approach to structural health monitoring. In: 2015 International Joint Conference on Neural Networks. New York (NY): IEEE; 2015.
[17] Iida F, Pfeifer R. Sensing through body dynamics. Rob. Auton. Syst. 2006;54:631–640.

[18] Hauser H, Ijspeert AJ, Fchslin RM, et al. The role of feedback in morphological computation with compliant bodies. Biol. Cybern. 2012;106:595–613.
[19] Escalona-Moran MA, Soriano MC, Fischer I, et al. Electrocardiogram classification using reservoir computing with logistic regression. IEEE J. Biomed. Health Inf. 2015;19:892–898.
[20] Konkoli Z. The state weaving environment-echo tracker (sweet/record-it) sensing setup and algorithm. USPTO, patent pending, 62/319972. 2016.
[21] Kantz H, Schreiber T. Nonlinear time series analysis. Cambridge: Cambridge University Press; 2003.
[22] Wiener J, Hale J. Ordinary and delay differential equations. Vol. 272, Pitman research notes in mathematics series. Essex: Longman Scientific and Technical; 1992.
[23] Suresh R, Srinivasan K, Senthilkumar DV, et al. Dynamic environment coupling induced synchronized states in coupled time-delayed electronic circuits. Int. J. Bifurcation Chaos. 2014;24:1450067/1–16.
[24] Zoran K. A perspective on Putnam's realizability theorem in the context of unconventional computation. Int. J. Unconventional Comput. 2015;11:83–102.
[25] Tour JM, Zandt WLV, Husband CP, et al. Nanocell logic gates for molecular computing. IEEE Trans. Nanotechnol. 2002;1:100–109.
[26] Kubo R. Statistical-mechanical theory of irreversible processes. I. general theory and simple applications to magnetic and conduction problems. J. Phys. Soc. Japan. 1957;12:570–586.
[27] Appeltant L, Soriano MC, SandeVan der G, et al. Information processing using a single dynamical node as complex system. Nat. Commun. 2011;2:468 (6pages).
[28] Appeltant L, Soriano MC, SandeVan der G, et al. Computing using delayed feedback systems: towards photonics. In: Eggleton BJ, Gaeta AL, Broderick NG, editors. Nonlinear optics and applications VI. Vol. 8434, Proceedings of SPIE. Bellingham: SPIE-International Society for Optical Engineering; 2012. p. 84341W.
[29] Soriano MC, Ortin S, Keuninckx L, et al. Delay-based reservoir computing: noise effects in a combined analog and digital implementation. IEEE Trans. Neural Netw. Learn. Syst. 2015;26:388–393.
[30] Wu AL, Zeng ZG. Dynamic behaviors of memristor-based recurrent neural networks with time-varying delays. Neural Netw. 2012;36:1–10.