Physical reservoir computing in a soft swimming robot

Yuta Horii\textsuperscript{1}, Katsuma Inoue\textsuperscript{1}, Satoshi Nishikawa\textsuperscript{1}, Kohei Nakajima\textsuperscript{1}, Ryuma Niiyama\textsuperscript{1}, and Yasuo Kuniyoshi\textsuperscript{1}

\textsuperscript{1}The University of Tokyo, Tokyo 113-8656, Japan
horii@isi.imi.i.u-tokyo.ac.jp

Abstract

In recent years, swimming robots have been developed to achieve efficient propulsion and high maneuverability that are possessed naturally by fish. Previous studies have attempted to achieve swimming similar to fish by control based on physical models and top-down architectures, but have encountered problems due to the high complexity of the underwater environment. Several research works have tried to overcome these problems by exploiting embodiment—that is, by mimicking the physical properties of fish. To achieve more intelligent swimming from the perspective of the embodiment, we focused on a framework called physical reservoir computing (PRC). This framework allows us to utilize physical dynamics as a computational resource. In this study, we propose a soft sheet-like swimming robot and a PRC-based architecture that can be used to emulate swimming motions by exploiting its own body dynamics for closed-loop control. Through experiments, we demonstrated that our system satisfies the properties required for learning swimming motion through supervised learning. We also succeeded in robust motion generation and environmental state estimation, opening up future prospects for more intelligent robot control and sensing.

I. Introduction

Regarding underwater swimming, a screw propeller is the most common propulsion mechanism. However, it is known that the propulsion efficiency, maneuverability, and environmental adaptability of this system is inferior to those of natural fish. Many swimming robots have been developed to realize swimming capabilities similar to fishes using engineering approaches (Aminur et al., 2018). Many of these robots have adopted methods such as controlling multiple actuators using physical models (Lighthill, 1960) based on the theories of fish swimming (Ding et al., 2013; Yu et al., 2004), or introducing hierarchical architectures that switch behavior top-down based on sensor information (Liu et al., 2006, Wang et al., 2019). However, dynamic control in a liquid environment is in general very challenging for conventional control schemes (see, for example, (Borlaug et al., 2019) for recent attempts). Meanwhile, Beal et al. showed that fish might solve such difficult problems bottom-up by using their own body as a computational resource (Beal et al., 2006).

They found that a dead fish can passively produce swimming motion in vortex wakes. Focusing on this idea of embodiment, other studies have achieved swimming with high efficiency and high maneuverability by adopting soft materials and mimicking the physical properties of fish (Salumae and Kruusmaa, 2011, Takada et al., 2010, Ziegler et al., 2005). However, these researches still have room for improvement in terms of high environmental adaptability and complex decision-making ability that fish possess naturally. We think that to achieve adaptive motion learning/generation—a task that usually requires complex information processing—it is important to come up with a novel scheme that can further exploit natural dynamics of soft body as a part of the control architecture.

To realize such system, we focused on physical reservoir computing (PRC), a machine learning framework that uses physical dynamics as a computational resource (Nakajima, 2020). In the field of PRC, the dynamics of physical components such as octopus arms made of silicone (Nakajima et al., 2013, 2014, 2015, 2018, Judd et al., 2019), a flexible spine of quadruped robots (Zhao et al., 2013), soft wings of a flying robot (Tanaka et al., 2021), spring-mass systems (Hauser et al., 2012), and tensegrity structures (Caluwaerts et al., 2013) have been exploited to achieve emulation tasks of complex nonlinear dynamical systems, or feedback control of the component itself. Reservoir computing (RC), which is the premise of PRC, is a framework for recurrent neural network (RNN) training that consists of an input layer, an intermediate layer called the reservoir, and an output layer. RC is characterized such that training can be performed by updating only the readout weights, while the weights inside the reservoir are usually fixed during training. Owing to this setting, any dynamical system, including physical dynamics, that has appropriate properties, such as the separation property (SP), approximation property (AP), and echo state property (ESP), can be used as a reservoir. SP and AP are common functions in RNNs, requiring the system to distinguish different inputs and map them correctly (Maass et al., 2002). ESP requires the output of the reservoir to depend only on the previous input sequence and not
on its initial state (Jaeger, 2001; Yildiz et al., 2012). Thus, a system that satisfies the ESP has a state forgetting property. However, these conditions do not guarantee that the system works sufficiently for the desired task (Schrauwen et al., 2007). Therefore, it is important to conduct physical experiments to check feasibility. In this study, we aim to show that the swimming system, including dynamics in water and body, meets the specified conditions and can be used as a computational resource in motion emulation as well as in environmental state estimation. Thus, we made a sheet-like swimming robot and conducted experiments.

In section II, we explain the design of our robot and its learning/control rules. In section III, we discuss the results of the experiment on learning swimming motions and generating them through closed-loop control in a fixed environment. In section IV, we discuss the results of the experiment on environmental state estimation while generating motion through an open-loop control. In section V, we discuss the results of the experiment on switching closed-loop control-driven motion depending on the environmental states.

II. Robot and training/control rule

Robot design

The robot we used is illustrated in Fig.1. It sways a silicone rubber sheet with a servomotor to produce propulsive force. The waterproof RC servomotor HS-646WP from Hitec Multiplex Japan, Inc. was adopted. Eight bend sensors (four pairs, MB060-N-221-A02 from Taiwan Alpha Electronic Co., Ltd.) were embedded in the rubber sheet. The sensor can only detect one side of the bend, so two sensors facing the opposite side were used as a pair. Fig.2 shows the details, with respect to the numbers assigned to each sensor on the rubber sheet, sensors 0 and 1, 2 and 3, 4 and 5, and 6 and 7 are installed in the same position along the lengthier side of the sheet, but are facing sides opposite to each other. The number of sensors we used is the largest number possible under manufacturing restrictions. The resistance of the bend sensor changed in response to the bending. This change in resistance is detected through the change in potential difference between the sensor terminals; however, because the change in resistance during operation is small, the voltage change was amplified using a Wheatstone bridge circuit and the instrumentation amplifier INA126 from Texas Instruments, Inc. The initial value of resistance is different for each sensor, so one of the resistors in the Wheatstone bridge circuit was made semi-fixed and adjustable. We also installed a semi-fixed resistor to adjust the gain of the amplifier. These semi-fixed resistors were adjusted manually by using screwdrivers at the beginning of each experiment. This procedure was needed since the resistance of the sensors we used altered through time. The change in resistance during one experiment was ignorable enough. As a controller, the Teensy® 4.0 Development Board from PJRC.COM, LLC. was adopted.

![Figure 1: The proposed swimming robot. It consists of a servomotor, a plastic head case, a plastic joint, a rubber sheet, and seven plastic floats. Plastic parts are made by 3D printing. Floats are attached to the rubber sheet to prevent the sheet from sinking below the water surface.](attachment:image1.png)

![Figure 2: The rubber sheet in the robot. Made by silicone casting using Dragon Skin 10 fast from Smooth-On, Inc. Eight bend sensors are embedded within.](attachment:image2.png)

Learning rule and feedback control

Based on the general setup of RC (Schrauwen et al., 2007; Lukoševičius and Jaeger, 2009), we implemented supervised learning. The architecture we used is illustrated in Fig.3. The reservoir dynamics can be expressed using the following equation:

\[
x(t) = h(r(t)),
\]

\[
x(t + 1) = f(y(t), r(t), q(t)),
\]

where \( r(t) \) is the reservoir state, \( y(t) \) is the target motor command, and \( q(t) \) is the environmental state that we controlled externally in the later experiments via a lever that is connected to the robot by a string. The precise definition of \( q(t) \) and its setup will be explained in section IV. \( x(t) \) is the partial observation of \( r(t) \) expressed as a vector of sensor outputs. In this study, we defined the reservoir as an entire system consisting of a robot and the surrounding water. This is because in the case of swimming, there is always a mutual interaction between the robot and water, as can be seen in the figure. In addition, the fluid environment itself is a highly complex network, making it a desirable computational resource, as seen in the original concept of a “liquid computer” (Natschläger et al., 2002).
Open-loop training  
Closed-loop motion generation

Water tank

Environmental state

Target motor command

Estimated motor command

Sensory outputs

Interaction between the robot and surrounding water

Update b, W

Fixed b, W

b

Figure 3: The architecture used for learning and generating motion. While training (shown in green), the system is driven by open-loop control using the target motor command. b and W are learned by linear regression. While in motion generation (shown in blue), b and W are fixed, and the system is driven by closed-loop control using the estimated motor command.

The readout function in our architecture is a simple linear combination, as shown in the following equation:

$$\hat{y}(t+1) = Wx(t+1) + b,$$

where \( W \) is the weight matrix and \( b \) is the bias, both derived via linear regression using time series data of \( x(t) \) and \( y(t) \). \( \hat{y}(t) \) is the estimated motor command. Learning was not performed in real time, but used data accumulated over a certain period of time in a teacher-forcing condition. The exact way in which data was accumulated will be explained in the next section. After learning, motion generation was performed by closed-loop control using \( \hat{y}(t+1) \) calculated from the sensor output \( x(t) \) of the previous timestep as an input signal instead of \( y(t+1) \).

III. Experiment on emulating swimming motions in a fixed environment (closed-loop)

To investigate whether our system satisfies the necessary conditions for emulating swimming motions, and whether there are differences in performance depending on the motion to be learned, we conducted an experiment in which swimming motions were learned and generated in a fixed environment. The experiment was conducted using the following procedure:

1. Obtain the applied motor commands and the corresponding sensor outputs for 6000 timesteps.
2. Learn weights by linear regression using the training data obtained above.
3. Drive the robot with closed-loop control using the weights obtained above, and thereafter acquire data.

The robot was loosely fixed with three strings as shown in Fig[4]. The target motor command \( y[°] \) was given as a sinusoid as shown in the following equation:

$$y(t) = A \sin \left( \frac{2\pi t}{T} \right) + \varphi$$

(4)

The offset \( \varphi[°] \) was fixed at a value such that the position of the sheet was at the center of the robot head when \( t = 0 \). Time \( t \) was given as a timestep rather than a continuous value. The length of timestep \( \tau \) was fixed at 10 ms. The variable parameters in the experiment were wavelength \( T[\text{ms}] \) (600-1100 ms) and amplitude \( A[°] \) (30-50°). Sensor outputs were acquired five times during one timestep, at 2 ms intervals, and the average was used as the sensor output during that timestep.

Five training datasets and one test dataset were obtained, as shown in Fig[5]. For each training dataset, estimation was performed for the test data using the learned weight \( W \) in an open-loop manner, and thereafter the accuracy was evaluated by the coefficient of determination \( R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \). The \( W \) with the largest \( R^2 \) was extracted to be used for motion generation using closed-loop control. The largest \( R^2 \) values are shown in Fig[6]. The closer \( R^2 \) is to 1, the higher the accuracy. In most conditions, the value is greater than 0.99, indicating that estimating the motor command while driven by open-loop control can be achieved with a high accuracy. However, this does not guarantee that the closed-loop control will succeed. The
Figure 5: Five training datasets each include data from 800 consecutive timesteps. The test dataset is a combination of five data from 400 consecutive timesteps each. The reason for extracting the test data from multiple locations this way is to prevent the test data from being too close to a particular training data. The test data was used for evaluating the five obtained sets of weights in an open-loop manner. The set of weights with the best performance was used for closed-loop control. One set of weights includes eight weights and one bias, corresponding to the number of sensors.

system driven by closed-loop control must possess not only a high emulating ability but also robustness in order to maintain motion in a noisy underwater environment.

Fig. 7 shows the time variation of the sensor outputs and motor commands during the closed-loop control using \( W \) obtained by learning, with the target motion set as \((T, A) = (600, 30)\). Both the sensor outputs and motor commands show periodic waveforms.

For the other \((T, A)\) conditions, the estimated motor command was analysed by curve fitting. A sinusoid was fitted to the time series data of \( \dot{q} \), and the average wavelength and amplitude \( T' \) [ms] and \( A' \) [°], respectively, of the fitting curve were obtained. The errors of \( T' \) and \( A' \), compared to \( T \) and \( A \), respectively, are plotted in Fig. 8. The grayed-out areas in the figure are the data that were excluded from the graph as incomparable because \( T' \) was greater than 3000 ms or \( A' \) was less than 3°. The incomparable motion was observed when \( T \) is large, and \( A \) is small. Here, the speed of motion, proportional to \( A/T \), is small, meaning that the reservoir dynamics is small. In this case, it may be that the system does not have sufficient input separability to perform the task. This is a possible reason why closed-loop control failed under these conditions.

In contrast, in conditions where the system was able to generate a motion close to the target motion, it was highly probable that the system had the necessary conditions for learning. As for the wavelength, the error is at most approximately 10%, thus the accuracy is relatively high. This shows that the system can emulate a certain temporal task. However, for the amplitude, the error is at most approximately 73%. In the condition of \((T, A) = (800, 30)\), where the error is the largest, the system generates a motion with a larger amplitude than the learning motion. Because the target motion for condition \((T, A) = (800, 30)\) is a relatively slow motion, the nonlinearity of the system might not have been sufficient for accurate learning. However, the system was still able to produce a periodic motion, unlike in the grayed-out conditions. In light of these results, it may be possible to say that during motion generation for condition \((T, A) = (800, 30)\), and maybe even for other conditions around, too, the nonlinearity of the system is maintained by generating an “exaggerated” motion.

**Motion generation under manual perturbations**

Here, we consider the change in motion when an external perturbation is applied to the reservoir during closed-loop control. A manual perturbation was given to the rubber sheet, as shown in the upper figure in Fig. 9. The lower figure in Fig. 9 shows the change in the estimated motor commands.
Generating a triangle wave motion

The system was also able to emulate a triangular wave motion using the same architecture. The equation for the motion is as follows:

\[
y(t) = \begin{cases} 
  \frac{2At}{T} - \frac{A}{2} + \varphi & \left(0 < t \leq \frac{T}{2}\right) \\
  -\frac{2At}{T} + \frac{3A}{2} + \varphi & \left(\frac{T}{2} < t \leq T\right)
\end{cases}
\]  

The upper figure in Fig.10 shows the estimated motor commands during closed-loop control for the target motion set as \((T, A) = (600, 30)\). The lower figure in Fig.10 shows a sinusoidal counterpart for comparison. The produced motions exhibit the characteristic of each target waveform.

IV. Experiment on environmental state estimation (open-loop)

To investigate whether the robot can discriminate the state \(q\) of the environment through its motion, we conducted an experiment. The robot first performs a sinusoid motion generated by an open-loop control under different \(q_0\)s. The system then learns the weight \(W_q\) to estimate \(q\) from the sensor output through logistic regression. We used logistic regression for learning because the target output takes a binary value as we explain below, and logistic regression is a useful method
when estimating a binary value. In this experiment, \( q \) corresponds to the state of the string connected to the robot head (Fig. 11). Two states UP and DOWN were switched by using a servomotor driven lever connected to the string. This setup is expressing the case in which the robot is under the influence of an external force. In the DOWN state, the robot is swimming freely, and in the UP state, the robot is resisting an upward force. For logistic regression, \( q \) was defined numerically as follows:

\[
q(t) = \begin{cases} 
1 & \text{(UP)} \\
0 & \text{(DOWN)}, 
\end{cases}
\]

(6)

The equation for the logistic regression is as follows:

\[
h(t + 1) = \frac{1}{1 + \exp(-(W'x(t + 1) + b'))}.
\]

(7)

\[
\hat{q}(t + 1) = \begin{cases} 
1 & \text{if } (h(t + 1) > 0.5) \\
0 & \text{if } (h(t + 1) \leq 0.5), 
\end{cases}
\]

(8)

where \( h(t) \) is the assumption function and \( \hat{q}(t) \) is the estimated state. The \( W' \) and \( b' \) were to be learned. We used the L2 regularization, in which the regularization parameter was fixed to 100.

For both states UP and DOWN, one training dataset and one test dataset were acquired, both through 3000 consecutive timesteps. Thereafter the two training datasets and the two test datasets were combined. The learning and evaluation was done by using these combined datasets. The conditions \((T, A)\) chosen in this experiment were \((640, 40)\) and \((800, 50)\). These two motions have different \( T \) and \( A \), but the motion speed, proportional to \( A/T \), is the same.

As for \((T, A) = (640, 40)\), the estimation accuracy was 0.991, which can be taken as a high enough accuracy. However, as for \((T, A) = (800, 50)\), the accuracy was 0.529. Since the accuracy will be 0.5 for a completely random guess, this indicates that estimation has failed. We will discuss the cause of this problem, considering the properties that the reservoir must satisfy.

First, we examine whether differences in the environment are reflected in the sensor outputs. Under each condition \((T, A)\), Fig. 12 shows the waveform of the sensor output for each lever state. In both conditions \((T, A)\), the sensor output in the UP state has high-frequency noise compared to the DOWN state. A difference can also be seen in the waveform and offset. Thus, the sensor output sequences do not seem to lack information needed for distinguishing the states.

Next, we decided to determine what properties are not enough for successful emulation by seeing if we can compensate for this by changing the readout part. In the experiments above, we estimated the environmental state or motor command using sensor outputs from only one timestep. Here, we performed the same environmental state estimation, except that we used sensor outputs from the past timesteps. The results are presented in Fig. 12. In both conditions, the accuracy gets overall larger when more data from the past timesteps are used. As for \((T, A) = (640, 40)\), the accuracy reaches 1.0 at 52 timesteps. As for \((T, A) = (800, 50)\), the accuracy starts to increase at around 20 timesteps, reaches approximately 0.85 at around 80 timesteps, but the increasing stops from there. The reason why the accuracy increased can be explained by memory capacity and dimensionality. The system as a whole must pos-
scess sufficient memory capacity to distinguish past inputs. In this altered setup, the memory of the reservoir is supplemented by the computer. In addition, in the latter case, the dimensionality of the sensor outputs is increased by the computer. The rich information is important for labeling different outputs and mapping them—that is, to satisfy the AP. Thus, it is highly probable that in the former setup, the number of dimensions of the sensor outputs (eight in this case) was insufficient to satisfy the AP.

As a result, the architecture first proposed in this study did not sufficiently satisfy some properties, thus the readout part needs to be modified in order to perform accurate estimation. In actual fish, not all computational processing is done by the body, and processing in the brain also plays an important role in determining behavior. In the future, it will be necessary to find an appropriate balance between the role of the body and the role of the controller by trying different kinds of setups. In addition, given that the accuracy was very different between the two conditions with or without the readout modification, the type of motion used for estimation may also be a key factor for good estimation.

V. Experiment on motion switching depending on environmental states (closed-loop)

In the previous experiment, we showed that environmental state estimation is possible by setting appropriate readouts. However, estimation itself is not sufficient for swimming with environmental adaptability. There are two ways to realize adaptability. The simple idea is that the controller switches the readout weight parameter set depending on the estimation. However, this method requires the controller to divide the state space. To achieve fish-like intelligence in the real world by this method, a great number of sets of weight parameters would be needed, or it is not even possible. A smarter solution is to allow the body to demonstrate adaptability—that is, to realize switching of motion without changing the readout weight. In this experiment, we investigated whether the system could emulate different motions under different environmental states. The definition of $q$ was same as in the estimation experiment.

For conditions $(q, T, A) = (1, T_{UP}, A_{UP})$ and $(q, T, A) = (0, T_{DOWN}, A_{DOWN})$, the weights $W_{UP}$ and $W_{DOWN}$ for generating closed-loop control was
obtained in the same way as in the experiment in a fixed environment. Then, the training data used for obtaining \(W_{UP}\) and \(W_{DOWN}\) were combined and used to obtain weight \(W_{BOTH}\) through linear regression, a weight that is expected to demonstrate motion switching when used for closed-loop control. Using these three weights, closed-loop control was performed under both UP and DOWN states. The average wavelength \(T'\) and amplitude \(A'\) of the estimated motor commands are shown in Table 1. The conditions \((T_{UP}, A_{UP}, T_{DOWN}, A_{DOWN})\) chosen in this experiment were \((640, 40, 800, 50)\) and \((800, 50, 640, 40)\). We refer to them as Conditions 1 and 2.

Under Condition 1, \(T'\) when driven by \(W_{BOTH}\) is approximately 10% larger than \(T\) when UP and approximately 5% smaller than \(T\) when DOWN. The values are shifted in a way that fills the difference in wavelength between the two states, and the amount of change in motion due to the change in state is reduced, but the magnitude relationship is maintained. As for \(A'\), the error from \(A\) is within 5% in both states, which is a relatively high accuracy. These results alone seem to indicate that motion switching has been successful. However, under Condition 2, the change in the values of both \(T'\) and \(A'\) due to the change in the environmental state is less than 3%, which is almost no change at all. Looking at \(A'\) when driven by \(W_{UP}\) and \(W_{DOWN}\) at Condition 1, \(A'\) at DOWN is larger than \(A'\) at UP by about 10°, even though these cases use weights obtained from only one condition. Therefore, it seems that the observed motion switching was not the result of learning, but was just a manifestation of the characteristics of the original motion. In fact, the values of \(T'\) and \(A'\) driven by \(W_{BOTH}, W_{UP}\), and \(W_{DOWN}\), indicate that an intermediate motion between the two target motions was generated for each condition.

VI. Conclusion

Using the PRC framework, we succeeded in emulating swimming motions by closed-loop control for a soft sheet-like swimming robot. In addition, we were able to make the system distinguish between two different environmental states through motion by altering the readout part. These results indicate that the robotic swimming system satisfies the properties necessary for reservoirs within the scope of the experiments conducted in this study. It is also interesting to note that in some biological systems, such as in the nematode worm *Caenorhabditis elegans*, it was shown that a closed-loop control without a central pattern generator is used for swimming in different media with the same neural circuit (Berri et al., 2009), which is similar to our proposed control scheme.

In each experiment, we used simple control signals such as sinusoidal and triangle waves. However, fish can produce more complex and non-periodical motions. To find out if this system can generate such motions, further research would be needed.

### Table 1: The average wavelength \(T'\) and amplitude \(A'\) of the estimated motor commands using different weights for closed-loop control at different \((T_{UP}, A_{UP}, T_{DOWN}, A_{DOWN})\) conditions. The cells in gray are the conditions where values are not necessarily close to the target because the state during closed-loop control is different from that during training.

<table>
<thead>
<tr>
<th>Condition</th>
<th>(T') [ms]</th>
<th>(A') [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target</strong></td>
<td><strong>UP</strong></td>
<td><strong>DOWN</strong></td>
</tr>
<tr>
<td>Condition 1: ((640, 40, 800, 50))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(W_{BOTH})</td>
<td>705</td>
<td>758</td>
</tr>
<tr>
<td>(W_{UP})</td>
<td>632</td>
<td>638</td>
</tr>
<tr>
<td>(W_{DOWN})</td>
<td>760</td>
<td>806</td>
</tr>
<tr>
<td>Condition 2: ((800, 50, 640, 40))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(W_{BOTH})</td>
<td>767</td>
<td>749</td>
</tr>
<tr>
<td>(W_{UP})</td>
<td>834</td>
<td>841</td>
</tr>
<tr>
<td>(W_{DOWN})</td>
<td>654</td>
<td>645</td>
</tr>
</tbody>
</table>

In the final experiment, the desired motion switching was not observed. However, it has been confirmed that the behavior changes in some way depending on the environmental input. It is expected that meaningful behavior switching will be implemented in the future by changing the way the conditions are selected, the way the behavior is switched, and the learning method.

In addition, in this study, we did not focus on swimming speed, acceleration, or posture. In the future, it is expected that swimming optimization targeting these factors will be achieved through reinforcement learning.

Finally, we expect our proposed architecture to be applied to other soft swimming robots. Our architecture is characteristic in that it does not require knowledge of the sensor type and position, making the architecture highly versatile. By changing the robot shape and sensor types, it is highly probable that a more intelligent robot will be realized in the future.

### Acknowledgements

This work was supported by the JSPS KAKENHI Grant-in-Aid for Scientific Research on Innovative Areas “Soft Robotics” project under Grant Numbers JP18H05466, JP18H05472. This work was also partially based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).
References


