Behavior Switching Using Reservoir Computing for a Soft Robotic Arm

Tao Li, Kohei Nakajima, Matteo Cianchetti, Cecilia Laschi, and Rolf Pfeifer

Abstract— **Soft robots have significant advantages over traditional robots made of rigid materials. However, controlling this type of robot by conventional approaches is difficult. Reservoir computing has been demonstrated to be an effective approach for achieving rapid learning in benchmark tasks and conventional robots. In this study, we investigated the feasibility and capacity of the reservoir computing approach to embedding and switching between multiple behaviors in a on-line manner in a soft robotic arm. The result shows that this approach can successfully achieve this task.**

I. Introduction

Soft robotics represents one of the new trends and challenges in biologically inspired robots[1]. Traditionally, robots are made of rigid materials and resemble the articulated structure of vertebrates. Motivated by the fact that soft materials are ubiquitous in living creatures, new types of robots that adopt elastic elements in their construction have been developed in recent years [2]. Soft robots have potential advantages over traditional rigid ones in terms of morphological flexibility and interaction safety. Soft robots' flexibility means they could be used as, for example, search and rescue robots, which could crawl through rubble and squeeze into small spaces, and minimal-invasive operation devices. In this study, we take the extreme of softness and consider soft robots whose body or major functioning parts are constructed exclusively of elastic elements. Some encouraging instances of these extremely soft robots have been developed in the last few years [3], [4], [5]. However, their functions are mainly achieved by smart structure design while the controllers are either not addressed or use traditional methods. There are enormous challenges in controlling soft robots since soft materials exhibit highly complex and time-varying dynamics under actuation and the expansion of the applied forces is hard to predict in the structure [6], just to name several. Facing these difficulties, one could envision that traditional robot control methods based on rigid kinematics and dynamics are hard to apply to soft robots in general. There is still no efficient method tailored for controlling soft robots. In this study, we focus on achieving behavior switching of soft robots.

The octopus is a good source of inspiration for learning a control strategy for soft robots. As an inspiration for soft robot construction, octopus arms are extremely compliant and exhibit complex dynamics. However, the octopus controls its soft arms flexibly and precisely to perform various behaviors [7], such as reaching [8], [9] for an object, catching it, and bringing it [10] to its mouth in a varying and often uncertain environment. Our previous study on an octopus-inspired soft robots control scheme proposed that timing, the time to initial motions, is an important factor in controlling soft robots [11], [12], [13], [14], [15]. To implement timing in a robot controller, recurrent neural networks (RNNs) are normally used. However, traditional supervised training methods for RNNs use gradient descent techniques and are subject to local minima, slow convergence, instability, and other limitations [16]. Reservoir computing [16], [17], [18] is a new way to construct and train the RNNs. In this approach, only the connection weights from the reservoir to the output nodes are trained; thus, many fast linear regression algorithms can be used. This characteristic makes it realistic to use reservoir computing in physical robotic platforms.

The main work of this study is as follows: (1) evaluate reservoir computing approach by using different type of sensors with distinct noise characteristic; (2) demonstrate that reservoir computing approach can be used to embed and switch among multiple sequential behaviors in a physical soft robotic platform; (3) and analyze the stability of reservoir to sensor noises. In this paper, we used the reservoir architecture called echo state network (ESN).

II. Experiment Setting

An experimental platform equipped with a soft robotic arm was built to evaluate the interaction among the controller, the soft body, and the environment. The platform setup, data acquisition, and experimental procedure are presented in this section.

A. Platform setup

The platform setup is shown in Fig. 1(a). It consists of a soft robotic arm, its actuation, sensing, control systems, and a water tank containing fresh water as the underwater environment. The soft robotic arm, which mimics the morphology of an octopus arm, is based on the prototype proposed in [19]. It is made of commercially available silicone rubber (ECOFLEX^{TM} 00-30), which has similar density and Young's modulus as the octopus arm [19]. The total length of the cone-shaped soft arm is 310 mm, with an actuated part of 80 mm, measured from the base. The rest 230 mm is passively driven. The actuated part has two nonextensible fishing cables embedded symmetrically to the center of the arm, as shown by the dashed lines in Fig. 1(b). Using two servo motors (Dynamixel^{TM} AX-12A+), the soft

* This work is supported by the European Commission in the ICT-FET OCTOPUS Integrating Project (EU project FP7-231608).

T. Li, K. Nakajima, and R. Pfeifer are with the Artificial Intelligence Laboratory, Department of Informatics, University of Zurich, 8050 Zurich, Switzerland, email: taoli@ifi.uzh.ch

M. Cianchetti and C. Laschi are with the Advanced Robotics Technology and Systems Laboratory, Scuola Superiore Sant'Anna, 56100 Pisa, Italy.

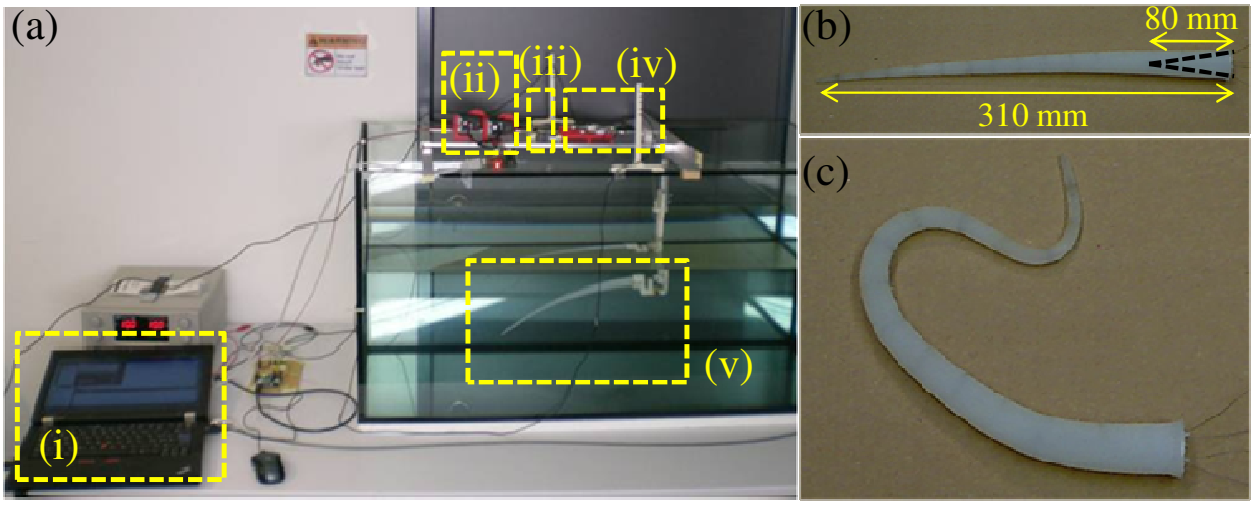

Fig. 1. (a) The experimental platform. It consists of a laptop PC (i), two servo motors (ii), one camera (iii), two force sensors (iv), and a soft robotic arm (v). (b) The soft robotic arm used in this paper. Dashed lines represent the cables embedded in the arm. (c) The robotic arm is made of silicone rubber and thus can be bent at any point and to any direction.

arm is driven by pulling the two cables embedded in the actuated part of the arm. The cable tensions are measured by two force sensors (KD24S from ME-Meβ system GmbH). The force sensor signals are amplified and sent to a PC serial port through an Arduino^{TM} UNO board, whose ADC outputs integer values between 0 and 1023, which correspond linearly to forces of 0 to 10 N. Thus, the unit of force sensor data is about 0.01 N. The force unit is designated by [POS] in this paper for clarity,. The servo motor positions are also sent to the PC as sensory inputs by integer values from 0 to 1023, which correspond linearly to angles of 0 to 300 degrees. The unit of position sensors is designated by [FCE], which is about 0.29 degrees. A camera (Logitech^{TM} Webcame Pro 9000) is placed on the top of the platform to record the the soft silicone arm motion.

A *Java* program running on a laptop PC receives the sensor signals explained above and sends out the motor commands to the servo motors. The unit of timestep, in this paper, is one sensing and actuation loop of the control program. Reservoir is prepared in the program to generate sensory-motor loop. For the sensor signals (S), it allows to take either the force sensor readings or the servo motor positions. The sensor selection depends on experimental setting explained below. We adopt three variables for the motor commands, which are the moving direction for each motor and their common speed. The overall settings of the reservoir and experimental procedures are explained in detail in the following sections.

The servo motor position, designated by integers between 0 and 1023, as described above, is adjusted according to the motor command (direction) and speed. The servo motor speed (v) is the motor position change per timestep and thus has the unit of [POS/t]. It can be set from 10 [POS/t] to 40 [POS/t] considering the limitation of the platform - a speed slower than 10 [POS/t] cannot exhibit the dynamics of the soft arm, while a speed faster than 40 [POS/t] would cause the servo motor to overheat. In this paper, motor commands are set as binary values, $M = \{+1, -1\}$. If the command gives $+1$ or -1, the motor is controlled to move from the current position toward the maximum position (L_{max}) or the relaxed position (L_{relax}) with the speed v, respectively. For each motor, L_{max} was determined so as not to cause the tip of the arm to touch the walls of the water tank during its movement. Note that the motor command does not always take the roller position to L_{max} or L_{relax}, but rather decides the motor moving direction for each timestep. Also, if the command gives $+1$ or -1 when the current position is in L_{max} or L_{relax}, respectively, then the position will stay unchanged for one timestep.

B. Reservoir Computing

1) Sensory - motor mapping: As we explained above, we prepared two types of sensors for the reservoir. The first type are force sensors (S_f) and the second are position sensors (S_p). Since the sensors measures two motors (cables), we describe them as $S1$ (S_f1 or S_p1) and $S2$ (S_f2 or S_p2). Meanwhile, since we aim to embed multiple behaviors into the network, we adopt control signals (C) as a input to the reservoir. Here, C is defined as a random real value in [0.0 1.0]. For example, if we want to control three behaviors, then we divide the range [0.0 1.0] equally and assign a control signal as (0.33, 0.66, 1.0) for each behavior. The correspondence between the assigned values and the behaviors is randomly determined and fixed in the training phase, as explained below. As a summary, we have $S1$, $S2$, and C as the input to the reservoir (Fig.2). For the outputs of the reservoir, we adopt previous explained motor commands (M) and speed (v). Since the two cables are controlled independently, there are two motor commands - $M1$ and $M2$. As a summary, there are three reservoir outputs in total (Fig.2).

2) Network architecture - Echo State Network (ESN): Fig.2 shows the ESN used in this study. It consists of four

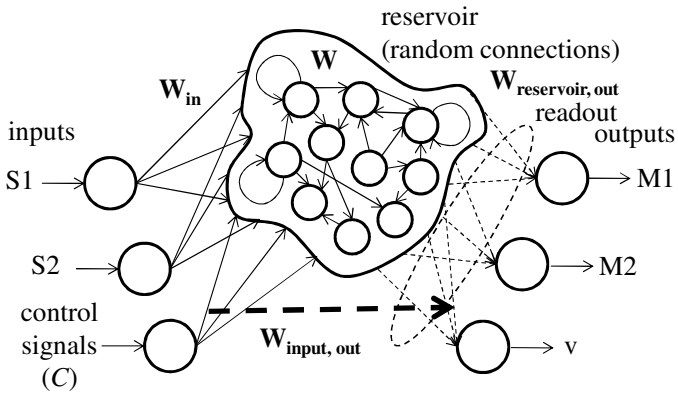

Fig. 2. Network architecture used in this paper. There are three input nodes (sensory inputs (*S*1, *S*2), and a control signal (*C*)), 200 reservoir neurons, and three output nodes (motor rotation directions (*M*1, *M*2), and motor speed (*v*)). All the connections are fully connected but not shown for simplicity. See text for details.

types of connection weights. The first type is connection weights from input nodes to the reservoir neurons (W_{in}, size 200×3). The second is connection weights that connect the reservoir neurons to each other (W, size 200×200). The third are direct connection from the input nodes to the output nodes ($W_{input,out}$, size 3×3). The last are connection weights from the reservoir neurons to the output nodes ($W_{reservoir,out}$, size 3×200). We use 200 neurons to construct the reservoir throughout this paper. Connection weights W_{in} are random real value from [0.0 1.0] and fixed throughout all the experiments. For the setting of W, we used the method introduced in [16], which can be summarized as the following procedure: (a). Randomly generate an internal weight matrix (W'). (b). Normalize W' to a matrix W'' with unit spectral radius by applying $W'' = W'/|\lambda_{max}|$, where $|\lambda_{max}|$ is the spectral radius of W'. (c) Scale W'' to $W = \alpha W''$, where $\alpha < 1$, whereby W has a spectral radius of α. α is set to 0.9, which is determined heuristically in this study. $W_{reservoir,out}$ and $W_{input,out}$ are the weights that will be adapted in the training phase. W_{out} is used to represent the concatenation of the two output weights matrix, $W_{reservoir,out}$ and $W_{input,out}$. $W_{out} := W_{reservoir,out} \oplus W_{input,out}$. We define input sequence in n timesteps as $u(n) = (u_1(n), u_2(n), u_3(n))$, where $u_1(n) = S1(n)$, $u_2(n) = S2(n)$, and $u_3(n) = C(n)$. The state of the neurons of the reservoir is $x(n) = (x_1(n), x_2(n), ..., x_{200}(n))$. The concatenation of the input sequence and neuron states is represented by $o(n) := x(n) \oplus u(n)$. The output of the network is $y(n) = (y_1(n), y_2(n), y_3(n))$, where $y_1(n) = M1(n)$, $y_2(n) = M2(n)$, and $y_3(n) = v(n)$. Then, the updating rules of the connection weights are defined as:

$$x^T(n+1) = g(W_{in} * u^T(n+1) + W * x^T(n)), \quad (1)$$

$$y^T(n+1) = W_{reservoir,out} * x^T(n+1) + W_{input,out} * u^T(n+1) \quad (2)$$

$$= W_{out} * o^T(n+1), \quad (3)$$

$$g(x) = \tanh(x). \quad (4)$$

(Note that, we actually applied $g(x)$ to calculate $y(n+1)$ because they are binary values.). Next, in order to determine W_{out}, we need to decide a teacher output ($d(n) = (d_1(n), d_2(n), d_3(n))$). As we will explain in the following section, $d(n)$ is determined according to behaviors we require. The internal states, $o(n)$ for $n = t_{start}$, $t_{start}+1$, ..., $t_{start}+t_{train}$ are collected into the rows of a state-collecting matrix X of size $t_{train} \times (200+3)$, where t_{start} is the timestep to start collecting the data and t_{train} are the timesteps used for the training data. At the same time, the teacher outputs $d(n)$ are collected into the rows of a matrix T of size $t_{train} \times 3$. Thus, the desired weights are directly obtained by multiplying the pseudoinverse of X (X^*) with T:

$$W_{out} = X^* T. \quad (5)$$

3) Experiment procedure and network training: The aim of this study is to evaluate whether the reservoir computing approach can be reliably applied to the physical soft robotic platform. When we try to embed control inspired by the octopus, we have to embed and combine various types of sequential control. As a preliminary exploration, we design the motor outputs for simple oscillatory behaviors of the robotic arm. By regulating the control signal, we aim to switch those behaviors in an on-line manner. Furthermore, we aim to explore the relation of the types of sensor (that is, the precise position sensors and noisy force sensors) to the reservoir performance. Since the position sensors take the values of servo motor position, they will not be affected by the dynamics of the soft robotic arm. However, the force sensors are expected to be strongly affected by the body dynamics of the soft arm, since they detect the forces on the cables embedded in the arm. We also try to evaluate the robustness to the noise of the reservoir.

The oscillatory behavior of the arm is achieved by alternatively adjusting the forces on the two cables embedded in the soft robotic arm. Initially, one cable (cable 1) is in its relaxed state and the other (cable 2) is in its maximum. Then, the motor driving cable 1 starts increasing the tension on the cable at a constant speed until the cable reaches the predefined maximum position, while cable 2 is driven moving toward the relaxation position. Then, cable 1, which is at its maximum position, starts to go back to the relaxed position, while cable 2 goes to its predefined maximum position. This alternative adjustment continues until a predefined timestep. There are three behaviors defined by different speeds. The three behaviors used in the experiment have speeds of 10 [POS/t] (behavior C), 18 [POS/t] (behavior B), and 26 [POS/t] (behavior A), corresponding to control signal of 1.0, 0.33, and 0.66, respectively. For each of the three behaviors defined in the experiment, the speed is the same for both cables and both directions.

To achieve the behavior switching, three phases are used: teaching, learning, and evaluating. In the teaching phase, the teaching data to be used to train the reservoir readout is generated for 8000 timesteps. A random control signal is generated at the beginning of every 200 timesteps, and the soft robotic arm oscillates at the corresponding speed.

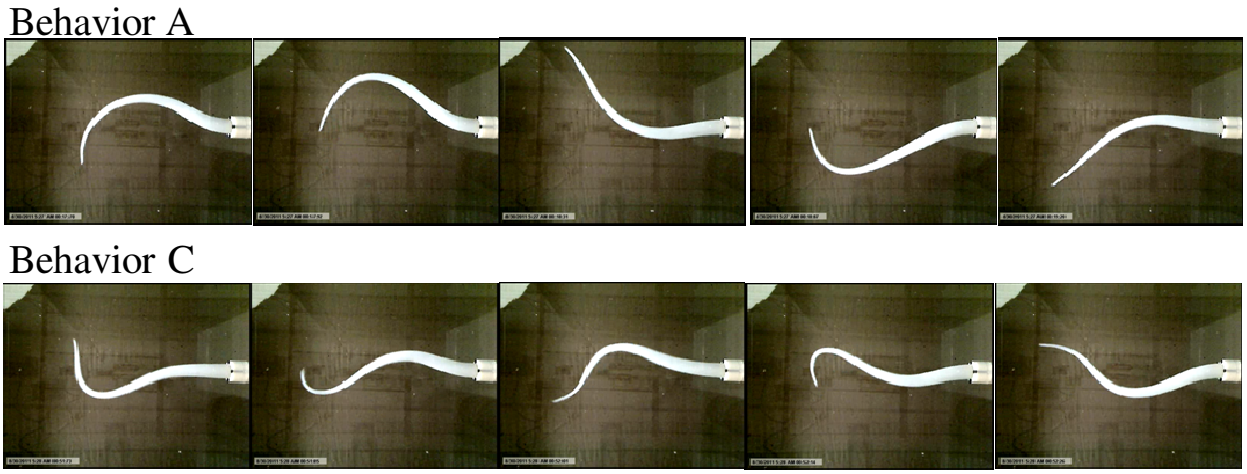

Fig. 3. Typical examples of the soft robotic arm behavior. The upper line shows behavior A, the lower line shows behavior C. Time evolves from left to right. We can see that the amplitude of the oscillatory behavior in behavior A is larger than in behavior C.

We record the control signal and the corresponding motor positions and forces. Then we use the teaching data generated in the teaching phase to train the linear reservoir readout connection weights. The first 200 timesteps teaching data is used to eliminate the effects of the arbitrary starting state and discarded as standard practice. In addition, a random noise is added to the sensor data to enhance the the reservoir's stability. The noise amplitude is determined by considering the sensor output range during the experiment. It is with an amplitude of 26 [POS] for the position sensors and 30 [FCE] for the force sensors. After training, the reservoir is implemented and evaluated to switch among the three behaviors for 5000 timesteps. First, we activate the arm using the same procedure as the teaching phase for 200 timesteps. Then, the reservoir takes a random control signal and generates the corresponding behavior. The generated positions $M(n)$ and the desired positions $d_m(n)$ for both cables at each timestep are recorded.

To evaluate whether the reservoir dynamics is necessary to generate the desired behavior switching, we also tested the setting without reservoir. This is essentially performed by setting the number of reservoir nodes $N = 0$. Therefore, only the connection weight from the input nodes to the output nodes are adapted during the training phase. Further experiments are carried out to analyze the robustness of the reservoir to position sensor noise. Position sensors are used in this experiment. Firstly, a set of training data are collected and used in all the training procedures in this experiment. In the learning phase, we use the same procedure. In the evaluation phase, 10 different levels of noise are added to the position sensor data. The 10 levels of random noise are set from 0 to 90 [POS] (the range of position sensor data is 185 [POS]) with an interval of 10 [POS]. Each noise level is tested 5 times. Error is evaluated by using the root mean square (RMS) of the errors in each timestep: $E_M = \sqrt{\frac{1}{t_{train}} \sum_{n=t_{start}}^{t_{start}+t_{train}} (d_M(n) - M(n))^2}$, $E_v = \sqrt{\frac{1}{t_{train}} \sum_{n=t_{start}}^{t_{start}+t_{train}} (d_v(n) - v(n))^2}$.

III. RESULTS

In this section, the behavior of the soft robotic arm is observed first. Then, the performance of the task to switch among three behaviors by the control signal is evaluated. We compare each case by first adopting the position sensors and then using the force sensors as input. Next, the performance of the controller without reservoir is evaluated to check the importance of the reservoir. Last, the stability of the reservoir is evaluated by adding noise to the position sensor input.

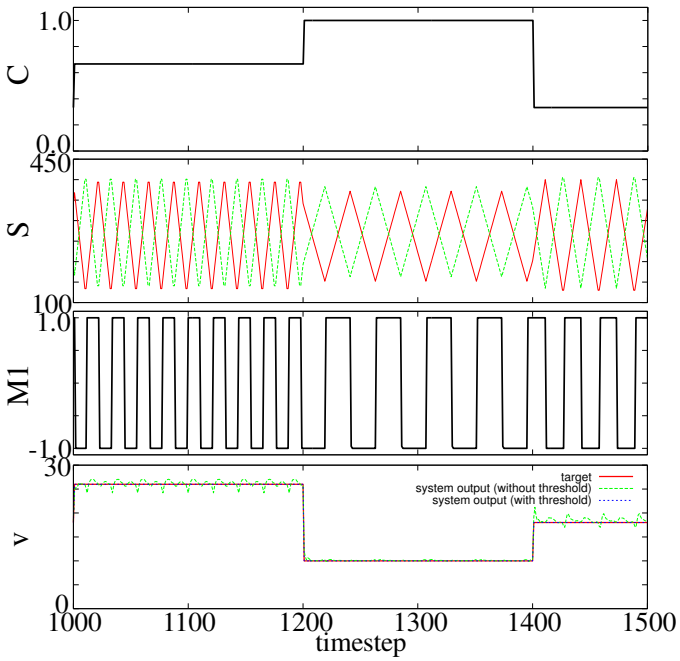

Fig. 5. The plots showing the timesteps from 1000 to 1500 in Fig.4. From the upper line to the lower line, it shows the trajectory of control signal (C), position sensor signal ($S1$)(unit: [POS]), motor command ($M1$), and speed (v)(unit: [POS/t]). In the plots, of the sensor signal, the red line shows $S1$ and the green line shows $S2$.

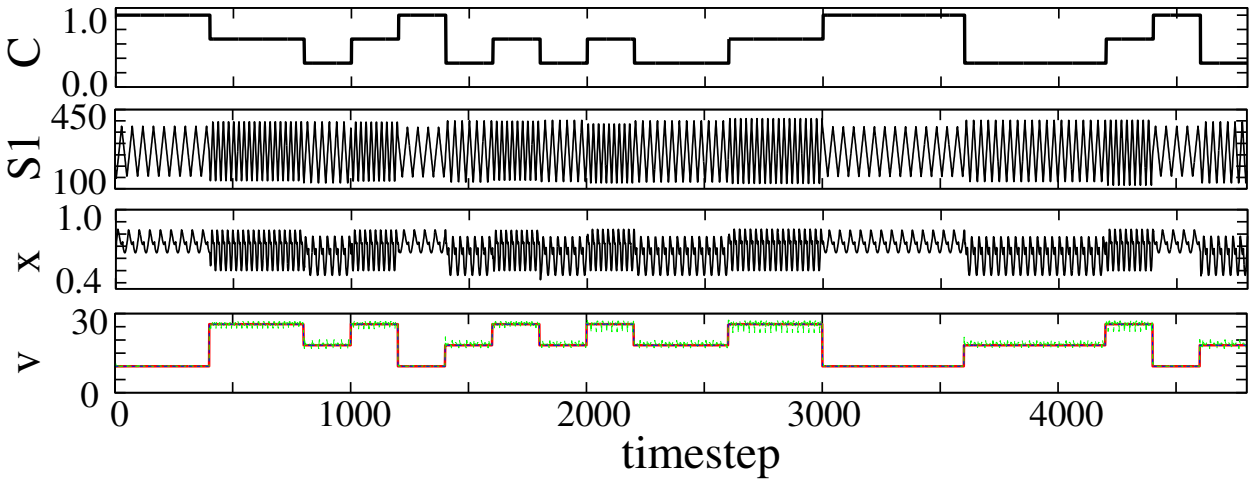

Fig. 4. Examples showing the trajectories of the variables when adopting the position sensors. From the upper to the lower line, it shows the trajectory of the control signal (C), position sensor signal ($S1$)(unit: [POS]), one reservoir neuron (x), and speed (v)(unit: [POS/t]). In the plots showing the trajectory of speed, the red line shows the target trajectory, while the blue line shows the output trajectory. They are overlapped in the plot. Note: units are not shown in the figure due to space limitation

A. Observations

The distinctive feature of a soft arm is the time delay to transmit the motion of the arm generated by the motors from the base to the tip since the arm is soft. This is an intrinsic feature of a soft body that is not observed in a rigid body, for example, a metal stick. As explained in the previous section, the oscillatory behavior is adopted and the speed of the arm oscillation is predefined from large to small in the order of behavior A, B, and C. The diverse behavior of the soft arm, which is controlled by the reservoir computing approach, is shown in Fig.3.

B. The influence of sensor type to reservoir performance

As mentioned in previous sections, the position sensors are not influenced by the diverse behavior of the soft arm as these sensors reflects the value of the angles of the servo motors. On the contrary, the force sensors are venerable to the effect of the dynamics of the soft body as it measures the forces on the two cables.

The behavior switching performance when the position sensors are adopted is shown in Fig.4. It can be seen that the dynamics of sensory data and a reservoir node, as an example, is switched clearly according to the control signal. Furthermore, the switching of the speed is achieved precisely. Fig.5 shows the details between the 1000 and 1500 timesteps. This figure shows that the pattern of the motor command is switched clearly according as the control signal. Next, let us check the case when the force sensors are adopted to reveal the influence of sensor type to reservoir performance. As shown in Fig.6, although the response of the sensors and the dynamics of the reservoir are switched by the control signal, a noisier pattern is observed compared with the case when the position sensors are used. Moreover, some errors can be observed in speed control. Fig.7 shows the details between the timesteps of 2500 and 3000 of Fig.6. Even the frequency of the motor command in each behavior is achieved to some extent, there are clear errors can be seen in the motor commands and motor speed.

C. The necessity of the reservoir

Fig.8 shows the results that no reservoir ($N = 0$) is used. It can be seen that neither the motor speed (v) nor the position sensor data ($S1$) has reached the desired values. The position sensor data ($S1$) is keep the same means motor 1 was not moving. Therefore, the reservoir dynamics is essential to achieve the behavior switching.

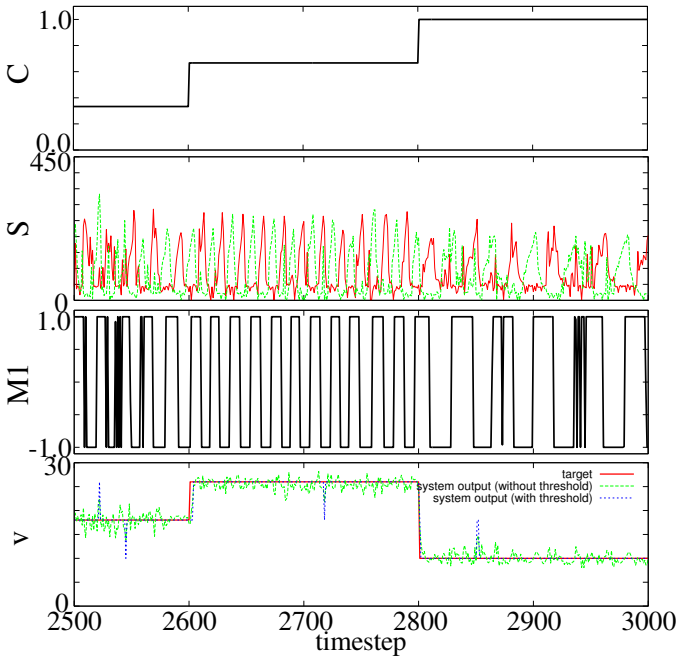

Fig. 7. The plots showing the timesteps from 2500 to 3000 in Fig.6. From the upper line to the lower line, it shows the trajectory of control signal (C), force sensor signal ($S1$)(unit: [FCE]), motor command, and speed (v)(unit: [POS/t]). In the plot of the sensor signal, the red line shows $S1$ and the green line shows $S2$.

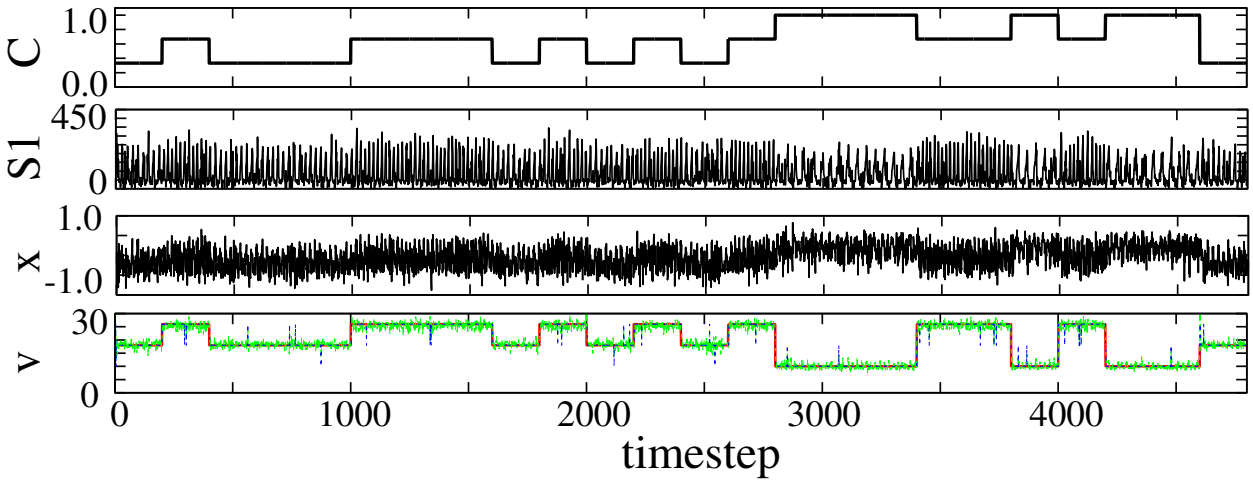

Fig. 6. Examples showing the trajectories of the variables when adopting the force sensors. From the upper to the lower line, it shows the trajectory of the control signal (C), force sensor signal ($S1$)(unit: [FCE]), one reservoir neuron (x), and speed (v)(unit: [POS/t]). In the plots showing the trajectory of speed, the red line shows the target trajectory, while the blue line shows the output trajectory. We can see that the output sometimes shows slightly different value from the target value. Note: units are not shown in the figure due to space limitation

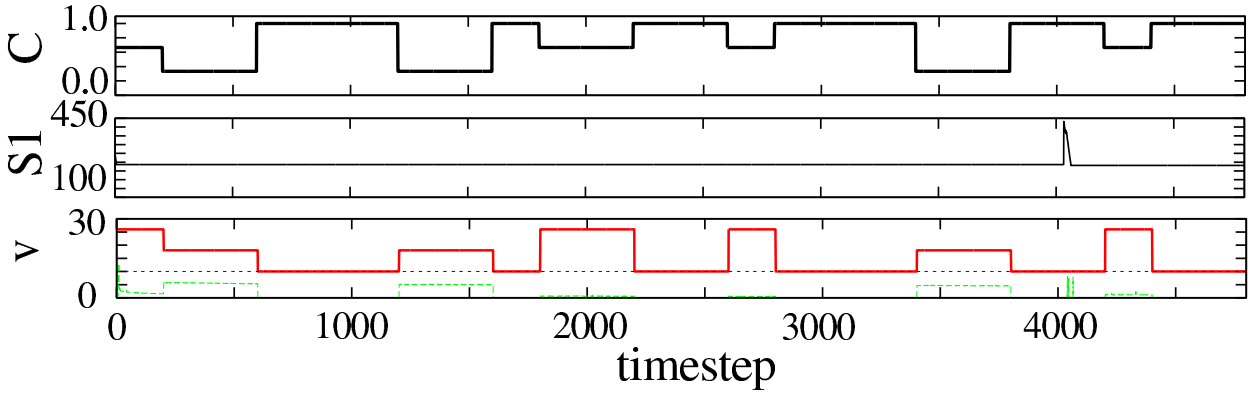

Fig. 8. Examples showing the trajectories of the variables when the reservoir was removed. From the upper to the lower line, it shows the trajectory of the control signal (C), position sensor signal ($S1$)(unit: [POS]), and motor speed (v)(unit: [POS/t]). In the plot showing the trajectory of speed, the red line shows the target trajectory, while the blue line shows the output trajectory. We can see that the output failed to achieve the desired values. Note: units are not shown in the figure due to space limitation

D. Stability to noise

One important criterion in evaluating a physical platforms controller is its robustness to noise. In this section, we estimate the noise impact on the task performance by adding noises to the position sensors data. First, we check the errors when adding different levels of noises to the position sensor data, shown in Fig.9. One can see a trend that both the RMS errors of motor commands and speeds increase with the increasing noise levels. This is consistent with the result shown in previous section that reservoir has better performance when using position sensors, which is not influenced by the soft robot body dynamics and shows less noisy data.

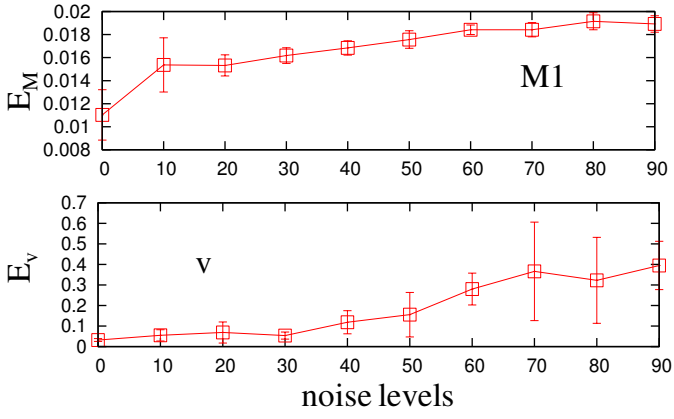

Fig. 9. Plots showing the errors according to the noise levels applied to position sensors. The upper line shows the RMS error plots of $M1$ (E_M), the lower one shows the RMS error of speed v (E_v)

IV. CONCLUSIONS AND DISCUSSIONS

This study shows that it is possible to switch multiple behaviors on-line using reservoir computing in a soft robotic platform. The overall performance was successful to achieve

periodic behaviors. Moreover, when two types of sensors were used (the position sensors and the force sensors), the performances differed. The performance was degraded when the force sensors were used compared with position sensors due to the body dynamics of the soft arm. In addition, when more behaviors were embedded, the performance changed. As the noise to the sensor data increased, the performance gradually degraded, but the performance was not affected so much by increasing the number of behaviors to 9. Finding a way to realize a more stable performance is a future objective.

In reservoir computing, it is possible to realize the simple and robust learning by adjusting only the readout in the training. But it is true that the performance also depends on portions other than the readout, such as the number of nodes and the spectral radius of the reservoir. As a matter of fact, the performance is also changed by the physical platform to be controlled, for instance, the type of sensor, as seen in this study. It is possible to enhance the robustness of the reservoir performance by looking into these issues.

Moreover, two additional aspects can be explored in future work. First, the control signal used to change among the behaviors can be more natural and realistic, for example, using visual sensors and different objects as stimuli. Second, The behavior used in this paper is a very simple periodic behavior. To be notified, the reservoir's performance depends on the task to be realized. In octopus, many interesting behaviors can be observed. For example, in reaching for an object, the octopus uses bending propagation in the arm, and the octopus forms a joint-like structure in the arm when fetching an object. Clearly, further improvements are needed to embed these behaviors. These aspects will be studied in our future work.

REFERENCES

[1] R. Pfeifer, M. Lungarella, and F. Iida, "Bio-inspired "soft" robotics: The new challenges ahead (periodical style - accepted for publication)," *Communications of the ACM*, to be published.

[2] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, 2008.

[3] E. Steltz, A. Mozeika, N. Rodenberg, E. Brown, and H. M. Jaeger, "Jsel: Jamming skin enabled locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, 2009, pp. 5672–5677.

[4] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, and H. M. Jaeger, "Universal robotic gripper based on the jamming of granular material," *Proceedings of the National Academy of Science*, vol. 107, pp. 18 809–18 814, 2010.

[5] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, 2011.

[6] X. Nie, B. Song, Y. Ge, W. Chen, and T. Weerasooriya, "Dynamic tensile testing of soft materials," *Experimental Machanics*, vol. 49 (4), pp. 451–458, 2009.

[7] T. Gutnick, A. R. Byrne, B. Hochner, and M. Kuba, "Octopus vulgaris uses visual information to determine the location of its arm," *Current biology*, vol. 21 (6), pp. 460–462, 2011.

[8] Y. Gutfreund, T. Flash, Y. Yarom, G. Fiorito, I. Segev, and B. Hochner, "Organization of octopus arm movements: A model system for studying the control of flexible arms," *Journal of Neuroscience*, vol. 16 (22), pp. 7292–7307, 1996.

[9] Y. Gutfreund, "Patterns of arm muscle activation involved in octopus reaching movements," *Journal of Neuroscience*, vol. 18 (15), pp. 5976–5987, 1998.

[10] G. Sumbre, G. Fiorito, T. Flash, and B. Hochner, "Neurobiology: Motor control of flexible octopus arms," *Nature*, vol. 433(7026), pp. 595–596, 2005.

[11] K. Nakajima, T. Li, N. Kuppuswamy, and R. Pfeifer, "Biologically inspired control of a simulated octopus arm via recurrent neural networks," in *Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO 2011)*. ACM press, 2011, pp. 21–22.

[12] ——, "Harnessing the dynamics of a soft body with "timing": Octopus inspired control via recurrent neural networks," in *Proceedings of the 15th IEEE International Conference on Advanced Robotics (ICAR 2011)*, 2011, pp. 277–284.

[13] ——, "How to harness the dynamics of soft body: Timing based control of a simulated octopus arm via recurrent neural networks," *Procedia Computer Science*, vol. 7, pp. 246–247, 2011.

[14] K. Nakajima, T. Li, H. Sumioka, M. Cianchetti, and R. Pfeifer, "Information theoretic analysis on a soft robotic arm inspired by the octopus," in *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011.

[15] T. Li, K. Nakajima, M. Kuba, T. Gutnick, B. Hochner, and R. Pfeifer, "From the octopus to soft robots control: an octopus inspired behavior control architecture for soft robots (periodical style - accepted for publication)," *Vie et Milieu/ Life and Environment*, to be published.

[16] H. Jaeger, "Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the "echo state network" approach," German National Research Center for Information Technology, Tech. Rep. 159, 2002.

[17] B. Schrauwen, D. Verstraeten, and J. V. Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks (ESANN2007)*, 2007, pp. 471–482.

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

[19] M. Cianchetti, A. Arienti, M. Follador, B. Mazzolai, P. Dario, and C. Laschi, "Design concept and validation of a robotic arm inspired by the octopus," *Materials Science and Engineering C*, vol. 31, pp. 1230–1239, 2011.