2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

A Bioinspired Synthetic Nervous System Controller for Pick-and-Place Manipulation*

Yanjun Li[1†], Ravesh Sukhnandan[2†], Jeffrey P. Gill[3], Hillel J. Chiel[3,4,5], Victoria Webster-Wood[2+], and Roger D. Quinn[1]

***Abstract*—The Synthetic Nervous System (SNS) is a biologically inspired neural network (NN). Due to its capability of capturing complex mechanisms underlying neural computation, an SNS model is a candidate for building compact and interpretable NN controllers for robots. Previous work on SNSs has focused on applying the model to the control of legged robots and the design of functional subnetworks (FSNs) to realize dynamical systems. However, the FSN approach has previously relied on the analytical solution of the governing equations, which is difficult for designing more complex NN controllers. Incorporating plasticity into SNSs and using learning algorithms to tune the parameters offers a promising solution for systematic design in this situation. In this paper, we theoretically analyze the computational advantages of SNSs compared with other classical artificial neural networks. We then use learning algorithms to develop compact subnetworks for implementing addition, subtraction, division, and multiplication. We also combine the learning-based methodology with a bioinspired architecture to design an interpretable SNS for the pick-and-place control of a simulated gantry system. Finally, we show that the SNS controller is successfully transferred to a real-world robotic platform without further tuning of the parameters, verifying the effectiveness of our approach.**

I. Introduction

Animals are capable of remarkable behavioral diversity, including locomotion and dexterous manipulation, that results from the coupling of their neural and motor systems [1], [2]. In the pursuit of robotic control methodologies that can capture this behavioral flexibility, various biologically inspired neural controllers have been proposed at different levels of biological realism [3]–[6]. Synthetic Nervous Systems (SNSs) are a promising approach to the control of robotic systems because (1) they incorporate elements of real neural dynamics, (2) the networks can be designed in a compact way to perform basic arithmetic operations, and (3) they have been used for real-time control of robots [7]–[10].

*This work was supported in part by the National Science Foundation (NSF) grant no. FRR-2138873 and by a GEM fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

[1]*Department of Mechanical Engineering, Case Western Reserve University, 10900 Euclid Ave, Cleveland, OH 44106, United States*

[2]*Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, United States*

[3]*Department of Biology, Case Western Reserve University, 10900 Euclid Ave, Cleveland, OH 44106, United States*

[4]*Department of Neurosciences, Case Western Reserve University, 10900 Euclid Ave, Cleveland, OH 44106, United States*

[5]*Department of Biomedical Engineering, Case Western Reserve University, 10900 Euclid Ave, Cleveland, OH 44106, United States*

[†]These authors contributed equally to the work

[+]Corresponding author vwebster@andrew.cmu.edu

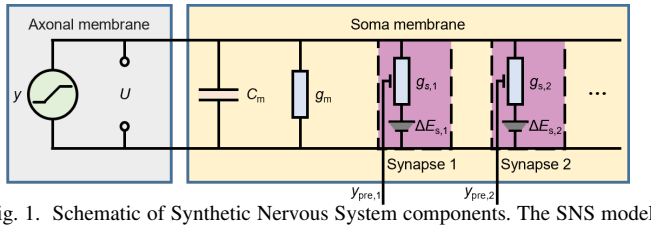

Fig. 1. Schematic of Synthetic Nervous System components. The SNS model incorporates membrane capacitance, leak conductance, and synaptic channels. The conductance currents govern the evolution of the membrane potential U. Instead of utilizing voltage-gated ion channels to explicitly generate spikes, the SNS model uses neural activity y to reflect temporal firing frequency.

SNSs have primarily been used in the past for the locomotion control in legged robots [7], [9], [11]. For example, the SNS approach has been used to successfully produce adaptive locomotion of the hind-legs of a dog-like robot [9]. SNS networks have also been used in the real-time control of the legs of the hexapod MantisBot for locomotion and steering [7]. However, SNS networks have not been previously demonstrated for grasping and manipulation control.

When designing complex controller architectures, one challenge for SNS networks is the number of parameters that must be tuned for stable performance. The Functional SubNetwork (FSN) approach has been used previously to tune the parameters of such SNS controllers by providing analytic constraints on the parameters of the arithmetic operations [8]. However, this approach requires analytical models of the subnetworks to perform optimization. To date, there has not been a systematic methodology that allows SNS network parameters to be learned for generic robotic motion control without using a closed-form analytical expression.

In this work, we address these gaps by demonstrating the design and implementation of a learning-based SNS grasping controller in a pick-and-place task. We (1) present a learning-based methodology for tuning the parameters of the SNS, (2) present a biologically inspired architecture for the development of SNS controllers for real-time robotic control, (3) expand the use of SNS controllers beyond locomotion and into the realm of real-time manipulation tasks, and (4) show that the SNS controller can be successfully trained in simulation and deployed on a real-world XYZ grasping system without additional parameter tuning.

II. Methodology

A. Discretization of Synthetic Nervous Systems

A Synthetic Nervous System (SNS) is a neural network (NN) inspired by neuroscience (Fig. 1). It considers synapses as ionic channels with different reversal potentials [12]. The

synaptic strength can be adjusted by changing the conductance. The governing equation of the ith neuron within an SNS can be written as the following ordinary differential equation:

$$C_{\mathrm{m},i}\frac{\mathrm{d}U_i}{\mathrm{d}t} = g_{\mathrm{m},i}(E_{\mathrm{r},i} - U_i) + \sum_j g_{ij}y_j\left(E_{ij} - U_i\right) + I_i \quad (1)$$

where U_i is the membrane potential of the ith neuron. $C_{\mathrm{m},i}$ is the membrane capacitance, $g_{\mathrm{m},i}$ is the membrane conductance, and $E_{\mathrm{r},i}$ is the neuron's resting potential. I_i models an applied external bias current. For the jth synapse, E_{ij} is the reversal potential, and g_{ij} is the maximal synaptic conductance. The product of g_{ij} and the activity of the jth presynaptic neuron y_j represents the synaptic channel conductance. A monotonically increasing activation function ϕ can be used to define the relationship between the output y_i and the state U_i. We set ϕ as a piecewise linear function $\mathbb{R} \to [0, 1]$:

$$y_i = \phi(U_i) = \frac{\min\left(\max(U_i, E_{\mathrm{lo}}), E_{\mathrm{hi}}\right) - E_{\mathrm{lo}}}{E_{\mathrm{hi}} - E_{\mathrm{lo}}} \quad (2)$$

The parameters E_{hi} and E_{lo} are the upper and lower threshold of the activation function, respectively[1].

Although SNS networks have previously been simulated as robotic controllers based on the differential governing equation [8], [9], the continuous time representation makes comparisons with existing classical artificial neural network (ANN) approaches and implementation in machine learning frameworks challenging. To address this challenge, we can obtain a discretized version of the SNS model by applying the semi-implicit Euler method [13], [14] to (1):

$$\begin{aligned} C_{\mathrm{m},i}\frac{U_i(t) - U_i(t-1)}{\Delta} &= g_{\mathrm{m},i}(E_{\mathrm{r},i} - U_i(t)) + I_i \\ &\quad + \sum_j g_{ij}\phi\left(U_j(t-1)\right)\left(E_{ij} - U_i(t)\right) \end{aligned} \quad (3)$$

where t is the current time and Δ is the time step. Letting $\tau_i = C_{\mathrm{m},i}/g_{\mathrm{m},i}$, $w_{ij} = g_{ij}E_{ij}/g_{\mathrm{m},i}$, $v_{ij} = g_{ij}/g_{\mathrm{m},i}$, $b_i = I_i/g_{\mathrm{m},i}$ and moving all terms containing t to the left hand side, we can further simplify (3) to

$$\hat{\boldsymbol{\tau}}_t = \frac{\boldsymbol{\tau}}{1 + \mathbf{V}\phi\left(\mathbf{h}_{t-1}\right)} \quad (4)$$

$$\mathbf{z}_t = \frac{\Delta}{\hat{\boldsymbol{\tau}}_t + \Delta} \quad (5)$$

$$\hat{\mathbf{h}}_t = \frac{\mathbf{b} + \mathbf{W}\phi\left(\mathbf{h}_{t-1}\right)}{1 + \mathbf{V}\phi\left(\mathbf{h}_{t-1}\right)} \quad (6)$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \hat{\mathbf{h}}_t \quad (7)$$

where $\mathbf{h}_t$ denotes the state vector $[U_1(t), \cdots, U_n(t)]^T$, $\boldsymbol{\tau}$ and $\mathbf{b}$ denote the time constant vector $[\tau_1, \cdots, \tau_n]^T$ and bias vector $[b_1, \cdots, b_n]^T$, respectively. The weight $\mathbf{W}$ and $\mathbf{V}$ satisfy $\mathbf{W}_{ij} = w_{ij}$ and $\mathbf{V}_{ij} = v_{ij}$, respectively. The operator $\odot$ denotes the Hadamard product (element-wise product).

Equations (4)-(7) imply that some classical ANNs are special cases of the SNS model. For example, when $\mathbf{V} = \mathbf{0}$, (4)-(7) degenerate to the semi-implicit Euler discretization of the continuous recurrent neural network (CTRNN) model [15]

$$\hat{\mathbf{h}}_t = \mathbf{b} + \mathbf{W}\phi\left(\mathbf{h}_{t-1}\right) \quad (8)$$

$$\mathbf{h}_t = \frac{\boldsymbol{\tau}}{\boldsymbol{\tau} + \Delta} \odot \mathbf{h}_{t-1} + \frac{\Delta}{\boldsymbol{\tau} + \Delta} \odot \hat{\mathbf{h}}_t \quad (9)$$

If we further set $\boldsymbol{\tau} = \mathbf{0}$, then (8)-(9) becomes the Vanilla recurrent neural network (Vanilla RNN), or a discretized version of the neural ordinary differential equations [16]:

$$\mathbf{h}_t = \mathbf{b} + \mathbf{W}\phi\left(\mathbf{h}_{t-1}\right) \quad (10)$$

When we study constant time series $\mathbf{h}_t = \mathbf{h}_{t-1}$, the time t is irrelevant and (10) is equivalent to a multilayer perceptron (MLP). Note that (4)-(7) have a similar form to the gated recurrent unit (GRU) [17], [18].

The network parameters to learn in SNSs are $\mathbf{W}$, $\mathbf{V}$, $\boldsymbol{\tau}$,and $\mathbf{b}$ in (4)-(7). The discretized version of SNSs can be trained by classical methods in RNN, such as backpropagation through time (BPTT) [19] for time series prediction problems.

B. Learning Nonlinear Operations with Discretized SNSs

Each conductance-based synapse in the SNS is parameterized by two parameters (g_{ij}, E_{ij} in (1) or $\mathbf{W}$, $\mathbf{V}$ in (4)-(7) that can be related to biological neural network measurements. In contrast, a synapse in ANN models is parameterized by a single parameter (synaptic weight $\mathbf{W}$ in (8) and (10)). The additional weight matrix $\mathbf{V}$ in the SNS allows the inputs to modify the apparent neuron time constants ($\hat{\boldsymbol{\tau}}_t$ in (4)) [14], [20], [21]. This mechanism is similar to update gates in GRU [17], which indicates the SNS has potential to learn time series dependencies. The incorporation of $\mathbf{V}$ turns the weighted sum term in (8) into a fractional function in (6), enabling the SNS to capture learning phenomena inspired by more complex dendritic integration [12], [22]–[24]. Such nonlinear interactions have potential for integrating multimodal information or performing conditional computation in robotic control [25].

To see how the rational dendritic integration term in the SNS increases the ability to perform such foundational calculations, we can consider a simple SNS with three neurons $\mathbf{h}_t = [U_{\mathrm{pre},1}, U_{\mathrm{pre},2}, U_{\mathrm{post}}]$ [8]. If we set $E_{\mathrm{lo}} = 0$, $E_{\mathrm{hi}} = 20$ and assume $U_{\mathrm{pre},1}$ and $U_{\mathrm{pre},2}$ are not saturated, then $\phi(U_{\mathrm{pre},1}) = U_{\mathrm{pre},1}/20$, $\phi(U_{\mathrm{pre},2}) = U_{\mathrm{pre},2}/20$. If $\mathbf{b} = \mathbf{0}$, $\boldsymbol{\tau} = \mathbf{0}$, $\mathbf{W}_{3,*} = [20,\ 0,\ 0]$, and $\mathbf{V}_{3,*} = [0,\ 20,\ 0]$, we have[2]

$$U_{\mathrm{post}}(t) = \frac{U_{\mathrm{pre},1}(t-1)}{1 + U_{\mathrm{pre},2}(t-1)} \quad (11)$$

When $U_{\mathrm{pre},2} \gg 1$, this network approximates division. The effect of $U_{\mathrm{pre},2}$ is similar to shunting inhibition in computational neuroscience. Shunting inhibition is key to neuromodulation [12], but it can not be easily achieved in classical ANNs.

We hypothesized that compact and sparse SNS networks can learn parameters for arithmetic. To test this hypothesis, we compared the ability of our SNSs and compact multilayer perceptrons (MLPs), a classical ANN model, to learn four

[1]Typical threshold values in an SNS network are $E_{\mathrm{lo}} = 0$ and $E_{\mathrm{hi}} = 20$

[2]The Asterisks notation represents a specific row in the matrix

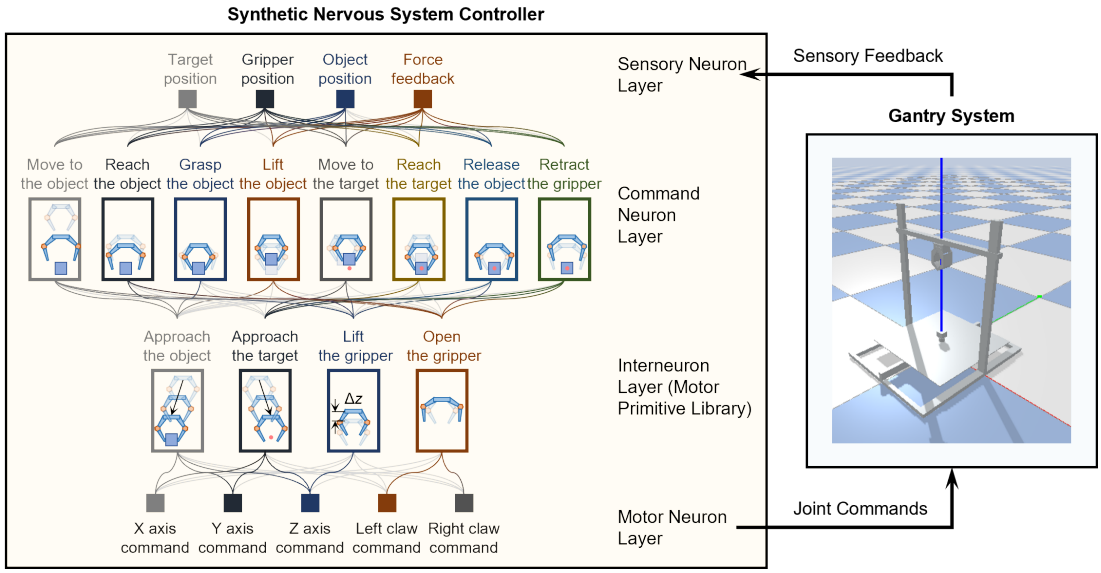

Fig. 2. Architecture for the SNS-based gantry control system. Each subnetwork in the command neuron layer and interneuron layer represents a neuron in the SNS controller. The command neuron layer contains 8 subnetworks whose activations represent eight different high-level commands for pick-and-place control. Each high-level command can be decomposed into subtasks existing in the motor primitive library. For example, the lifting the object command can be decomposed into approaching the object and lifting up the closed gripper. Similarly, each subnetwork in the interneuron layer represents a specific motor primitive which can be further decomposed into corresponding patterns of motor neuron activities. Therefore, subnetworks in the command neuron layer and the interneuron layer only need sparse connections (represented by colored lines) to selectively activate or inhibit related subnetworks in the lower layer. We grayed out the connections existing in the fully connected neural network but not existing in our architecture to emphasize the sparsity. The sensory neuron layer is responsible for collecting the sensory feedback and sequentially activating the subnetworks in the command neuron layer.

operations: addition, subtraction, division, and multiplication. The SNS for each operation is inspired by previous work on functional subnetworks [8]. We adopted similar architecture for MLPs (single layer, layer size 1 for addition, subtraction, and division; for multiplication, two layers, layer size (2,1)). We formulated the arithmetic operation learning as a time sequence regression task and adopted the mean square error as the training loss function. The training set includes 1000 training examples with two constant feature series and the corresponding label series. We set $\Delta = 0.1$sec and the sequence length as 50. The values of the constant feature series are randomly selected from $[E_{\text{lo}}, E_{\text{hi}}]$. We used Adam [26] as the optimizer. To compare learning performance between SNS and MLP networks, we compared training loss.

C. Designing and Learning Controllers with Discretized SNSs: Pick-and-Place Case Study

To assess the ability of our discretized SNS approach to learn robotic control parameters, we developed a grasper pick-and-place controller. The network architecture is abstractly inspired by that of feeding in the sea slug *Aplysia californica* [13]. Generally, the neurons responsible for *Aplysia* feeding can be organized into sensory neurons, motor neurons, interneurons, and command-like neurons. Command-like neurons receive feedback signals from sensory neurons. They control transitions between feeding behaviors by activating interneurons based on sensory feedback [27], [28]. The interneurons can be organized into subnetworks that realize essential control functions [29]. In our architecture (Figure 2), sensory signals for the target position, gripper position, object position, and force feedback are received by a sensory layer. These neurons are connected to a command layer to control behavioral switching, which projects onto an interneuron layer that coordinates motor primitives and sends signals to the motor neuron layer for actuator control. For each function in the network architecture (Fig. 2), the SNS controller (Fig. 3) possesses a corresponding neuron expressed as the discretized SNS described in section II-A. Network parameters are initialized and trained by the supervised learning paradigm in Section II-B so that subnetworks can generate the specified input-output relationships without using analytical methods like FSNs. Fig. 3 also demonstrates the synaptic polarity of connections found by the learning process.

D. Simulation Environment for Gantry SNS Parameter Tuning

To enable fine-tuning of the SNS before use in controlling a physical system (Fig. 4), a simulation environment was created in PyBullet (Fig. 2) [30]. This simulation environment captures the major features of the robot: (1) the dimensions of the gantry, grasper, and object are true-to-size, and (2) position control, where at each timestep a position command of where the system should be at the next timestep was sent to the simulation. The velocity limits of the axes were tuned to correspond with the peak velocities of the physical system.

E. Experimental Validation

A Creality CR-10 S5 3D printer [31] was modified to accommodate a servomotor actuated grasper to validate SNS control in a physical system (Fig. 4). This system could translate in three orthogonal axes (x, y, z), as well as control the angle between the claws of the symmetric grasper. The 3D printer's firmware was modified to control grasper state via M-code commands and query the 3D printer's stepper

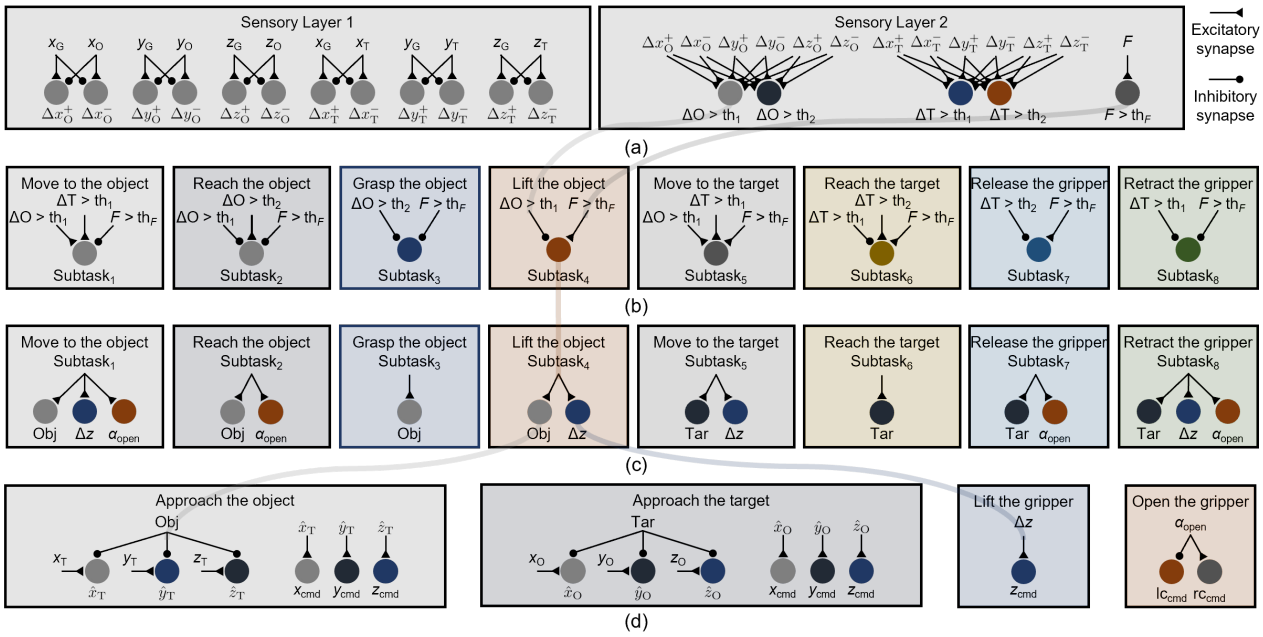

Fig. 3. The hierarchical SNS controller for pick-and-place control. (a) The first sensory layer calculates the difference in position between the object and the gripper ($\Delta x_{\mathrm{O}}, \Delta y_{\mathrm{O}}, \Delta z_{\mathrm{O}}$), and between the target and the gripper ($\Delta x_{\mathrm{T}}, \Delta y_{\mathrm{T}}, \Delta z_{\mathrm{T}}$) in the x, y, z axes. The second sensory layer receives inputs from the first layer and from force feedback to determine whether the distance is within a user-specified range bounded by (th_1) and (th_2), and whether the force is greater than a threshold (th_F). (b) The command layer contains 8 neurons ($\mathrm{Subtask}_{1-8}$) corresponding to 8 subtasks to accomplish. Neurons in this layer are activated if all presynaptic neurons with excitatory synapses are firing while those with inhibitory synapses are silent. (c) The interneuron layer contains four neurons representing four motor primitives: Obj (moving to the object), Tar (moving to the target), Δz (lifting up the gripper), and α_{open} (opening the gripper). Neurons in this layer are activated if any of the presynaptic neurons are firing. (d) The motor neurons layer contains motor neurons for xyz movement (xyz_{cmd}) and the angles of the claws of the gripper, ($\mathrm{lc}_{\mathrm{cmd}}$ and $\mathrm{rc}_{\mathrm{cmd}}$). These neurons receive transmission pathways [8] from their presynaptic neurons. We also include neurons receiving the object position ($\hat{x}_{\mathrm{O}}$, $\hat{y}_{\mathrm{O}}$, $\hat{z}_{\mathrm{O}}$) and the target position ($\hat{x}_{\mathrm{T}}$, $\hat{y}_{\mathrm{T}}$, $\hat{z}_{\mathrm{T}}$) in this layer. Neuron Obj has inhibitory modulation connections to $\hat{x}_{\mathrm{T}}$, $\hat{y}_{\mathrm{T}}$ and $\hat{z}_{\mathrm{T}}$, while Tar has inhibitory modulation connections to $\hat{x}_{\mathrm{O}}$, $\hat{y}_{\mathrm{O}}$ and $\hat{z}_{\mathrm{O}}$. The existence of these modulation pathways allows the SNS controller to select from moving to the object or moving to the target. The subnetworks highlighted by the blue borders help us walk through an example showing how a high-level command is triggered and implemented. If the gripper is close to the original position of the object ($\Delta\mathrm{O} > \mathrm{th}_1$ not activated) and the gripper has grasped the object ($F > \mathrm{th}_F$ activated), the command neuron corresponding to lifting up the object ($\mathrm{Subtask}_4$) starts firing. It in turn activates two interneurons Obj and Δz. Neuron Obj inhibits the sensing of the target position so that the motor neurons receive the original position of the object, while Δz increases the activity of z_{cmd} by a small amount. Therefore, the gripper can lift the grasped object.

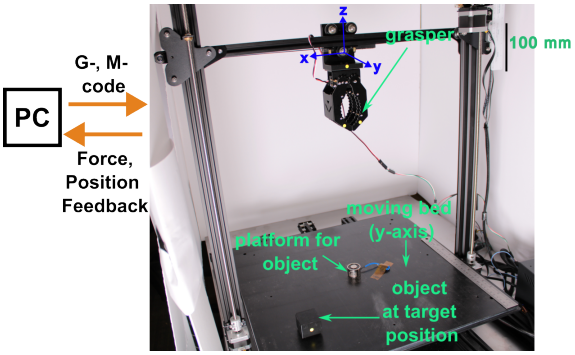

Fig. 4. Grasping robot used to validate SNS control. System is in the origin position of $(0, 0, 0)$. Object is at the target position. The grasper is mounted such that it can move in the $x - z$ plane. The platform moves in the y axis.

motor positions. Assuming no missed steps during actuation, this enabled pseudo-closed loop control of the gantry in the x, y, and z axes, as well as the grasper angle, $\theta_{grasper}$. Contact force information was provided by manually pressing a tactile pushbutton switch when contact was made between the grasper and the object and was registered by the gantry firmware.

The gantry position and force were inputs for the SNS (Fig. 2). At each timestep, the SNS controller uses these inputs to produce the next set of $(x, y, z, \theta_{grasper})$ target coordinates. The object position input was set as $(0, 0, -0.305)$ m, and the target position as $(0.15, 0.15, -0.310)$ m. The object was a 19.7 g cube of size $(39, 39, 34.5)$ mm.

III. Results and Discussion

A. SNS Networks can Learn Parameters for Key Mathematical Calculations better than Compact MLPs

Both the SNSs and compact MLPs specified in section II-B could learn the arithmetic operations for addition and subtraction (Fig.5(a-b)). The learning process of the MLPs is faster than the SNSs in these two cases because the operations to learn are linear functions, and the additional parameter $\mathbf{V}$ in SNSs is redundant. However, only the SNSs could learn close approximations to the division and multiplication operations (Fig.5(c-d)). Compact MLPs are insufficient to learn them because they lack inductive bias for nonlinear computation.

B. The SNS is Capable of Real-time Grasping Control

The SNS-controlled real-time, real-world grasper system (Fig. 4) was able to successfully recreate the behavior of the SNS-controlled simulation, namely: (1) grasp the object, (2) move the object to the target position, and (3) return to the starting position (Fig. 6). Importantly, the SNS controller whose parameters were tuned in the simulation environment

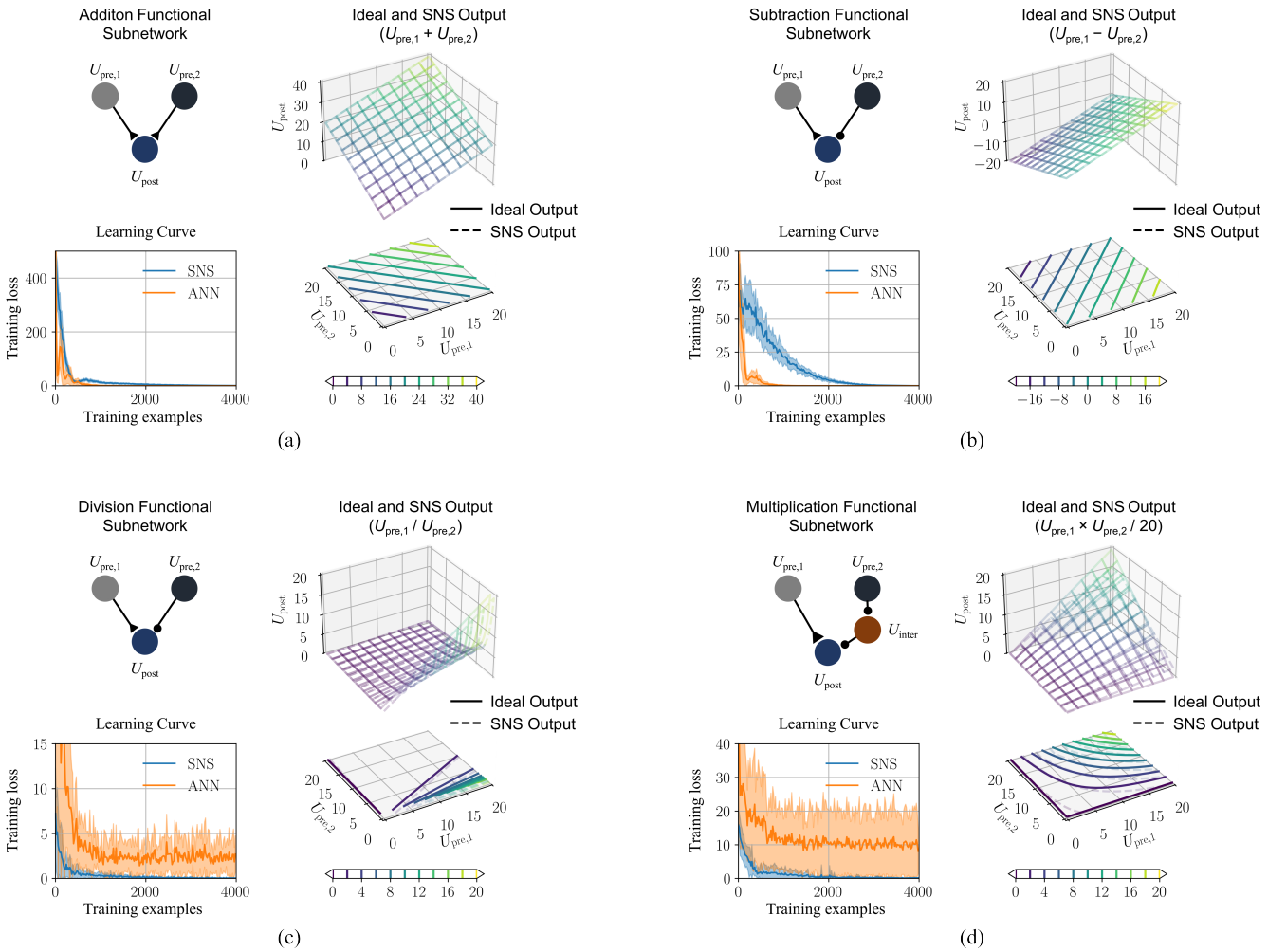

Fig. 5. Network diagrams, learning curves, functions, and contours of the ideal and SNS output for addition (a), subtraction (b), division (c), and multiplication (d). Triangular synaptic terminations represent synapses becoming excitatory after learning, and filled round terminations represent synapses becoming inhibitory after learning. In the multiplication learning task, we initialized the connection between the presynaptic neuron to the interneurons as an inhibitory synapse to avoid potential local minima. All parameters are initialized randomly for other tasks. The comparison between the ideal and SNS output and their corresponding contours shows that SNSs can perform these operations. The learning curves also suggest SNSs can learn multiplication and division quickly, while the compact ANNs cannot generate close approximations to these nonlinear operations.

was able to be used as-is in the control of the real grasper system without the need for any additional parameter tuning.

In addition to the SNS computation of the next commanded state $(x, y, z, \theta_{grasper})$, the controller also had to handle the communication with the gantry system. These I/O operations, where commands and state information were exchanged between the PC and gantry system, resulted in a longer average controller timestep for the real gantry system (85.5 ± 14.9 ms) when compared to the simulation (16.5 ± 4.2 ms). For both the real and simulated grasper systems, the average time taken to execute the SNS computations was comparable (5.9 ± 1.4 ms and 4.8 ± 2.2 ms, respectively). Future work will explore avenues for further improvement of the controller bandwidth by optimizing the computations performed in the SNS and in the I/O operations between the controller and grasper system.

It should be noted that the gantry essentially operates under position control as stepper motors are used to drive the x, y, and z axes. Hence, the position of the gantry could be directly controlled without considering the forces needed to produce such a motion, as long as the axes' peak acceleration and velocity limits were respected. Future work will explore the ability of the SNS to control robotic systems with more complicated dynamics, such as a 6-DOF robotic arm.

C. Kinematics of the Real-World Grasper System are Similar to those of the Simulation under SNS Control

Despite the increased controller timestep in the physical grasper system, the kinematics of the physical SNS-controlled grasper (Fig. 6e-h) are qualitatively similar to those of the simulated grasper system (Fig. 6a-d). Moreover, the physical system was capable of executing the behavioral patterns prescribed by the SNS's Command Neuron Layer (Fig. 2), which allowed the grasper to replicate the simulation's success in grasping and moving the object to the target location (Fig. 6i).

However, there were notable differences in the kinematics of the real-world physical system when compared to the simulation. In phases **III** to **IV** (Fig. 6i), the gantry grasps

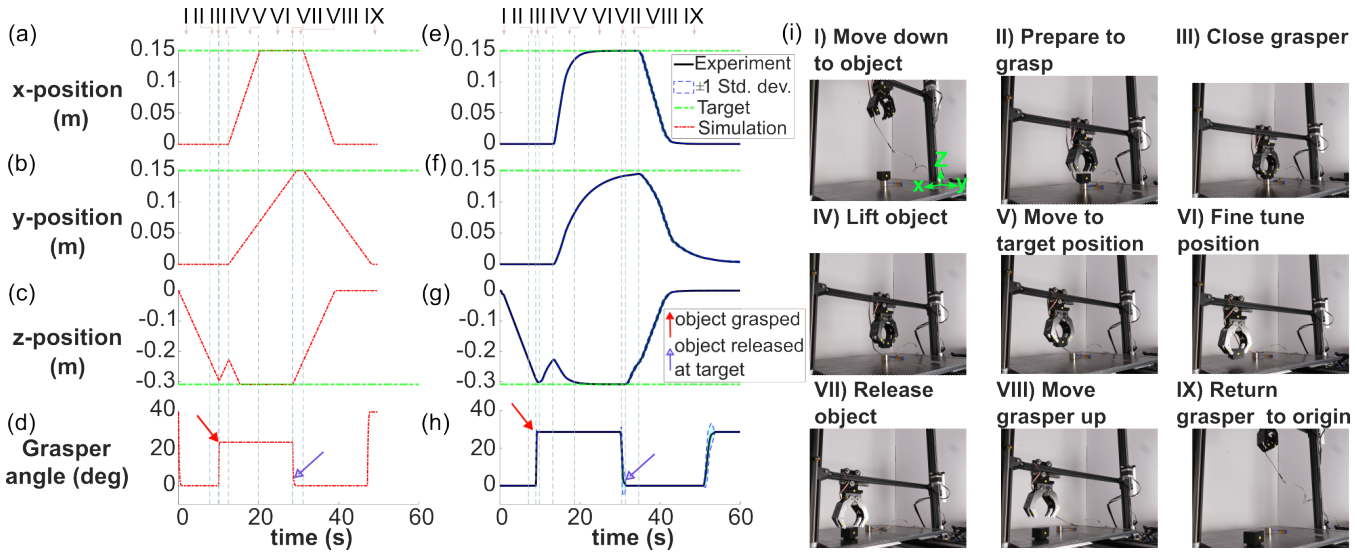

Fig. 6. (a), (b), (c) Kinematics of the x, y, and z axes, respectively, of the simulated gantry under SNS control. (d) Grasper angle kinematics. (e), (f), (g), (h) Kinematics of the x, y, z, and the grasper, respectively, of the real gantry under real-time SNS control. (i) State of the gantry, grasper, and object under real-time control. States **I**) to **VIII**) correspond to the dominant Command Neuron active at that point in time (Fig. 2). State **IX**) represents the return of the grasper to its original position. This behavior naturally comes about from the dynamics of states **I** to **XIII** by setting the object position to the origin.

the object and begins to ascend vertically, corresponding to the "Grasp Object" and "Lift Object" Command Neurons, respectively (Fig. 2). In the simulation, the transition from the descent of the grasper in the z axis to the ascent of the grasper occurs instantaneously as contact information can be determined immediately (Fig. 6c). In the physical grasper system, however, there is a small delay between these actions (Fig. 6g), as contact is determined by a manual press of a push-button. This is also the reason for the small delay between the release of the object at the end of phase **VII** (Fig. 6h) and the beginning of the grasper retraction in phase **VIII** (Fig. 6g). The simulation also maintained higher velocities in the y and $+z$ axes of the physical system. Possible reasons for the slower robot motions in those axes are that the y axis has the largest mass of all three axes, and the $+z$ axis must work against gravity. Further tuning of the simulation's dynamics may allow for a better match to the physical system.

Though not evident from Fig. 6e-h, the physical grasper system would intermittently exhibit non-smooth movement. This was most acute in the x-y plane during phases **V** to **VIII** when the difference between the commanded position and the current position was small because the gantry slowly approached the target position to release the object. The choppy motion observed during this behavior can be attributed to the longer than expected controller timestep and the trapezoidal velocity motion profile of the axes as they moved towards the commanded position. If the time taken to move from the current position to the commanded position was smaller than the time to execute the control loop, then such choppy motion could occur. To compensate for the increased overhead introduced by the I/O operations, moves were buffered on the controller side (maximum of 10 move commands) to improve motion smoothness. Future experiments will benefit from improving the controller timestep and optimizing the motion profile to prevent short-burst decelerations and accelerations.

IV. Conclusions and Future Work

Bioinspired approaches to the control of robots have the potential to enable the type of behavioral flexibility found in animals. In this paper, we have shown that bioinspired Synthetic Nervous Systems (SNSs) are a promising approach to robotic control as they have reduced training time for learning foundational mathematical operations in robotic control, such as division and multiplication, when compared to Vanilla ANNs (Fig. 5). We have also shown that SNSs can successfully control a robotic grasper to perform a pick-and-place manipulation task. We demonstrated that an SNS controller can be tuned in a simulation environment and subsequently used in a real physical system for a pick-and-place task without additional tuning (Fig. 6). The real grasper system's kinematics closely followed the simulation's behavior.

Ultimately, the use of supervised learning to determine SNS parameters (Fig. 5), and the hierarchical technique to create a pick-and-place controller (Fig. 3) described in this paper can be extended to other robotic systems to leverage the SNS's capability of performing foundational operations efficiently. Future work will validate the SNS real-time controller for use in controlling robotic systems with more complex dynamics, such as a 6-DOF robotic arm, or more complex perception, such as proprioceptive and haptic feedback to manipulate soft, fragile, and irregularly shaped objects. As the SNS includes elements of real neural dynamics, we will extend this bioinspired controller to include plasticity to realize real-time learning and adaptation [32], [33] which can automatically find a hierarchical and sparse controller for manipulation tasks.

Acknowledgment

We thank Kevin Dai for help in proofreading the manuscript. Funding to attend this conference was provided by the CMU GSA/Provost Conference Funding.

REFERENCES

[1] D. N. Lyttle, J. P. Gill, K. M. Shaw, P. J. Thomas, and H. J. Chiel, "Robustness, flexibility, and sensitivity in a multifunctional motor control model," *Biological Cybernetics*, vol. 111, no. 1, pp. 25–47, 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5326633/

[2] K. Nishikawa, A. A. Biewener, P. Aerts, A. N. Ahn, H. J. Chiel, M. A. Daley, T. L. Daniel, R. J. Full, M. E. Hale, T. L. Hedrick, A. K. Lappin, T. R. Nichols, R. D. Quinn, R. A. Satterlie, and B. Szymik, "Neuromechanics: an integrative approach for understanding motor control," *Integrative and Comparative Biology*, vol. 47, no. 1, pp. 16–54, Jul. 2007. [Online]. Available: https://doi.org/10.1093/icb/icm024

[3] G. A. Bekey, "Biologically inspired control of autonomous robots," *Robotics and Autonomous Systems*, vol. 18, no. 1-2, pp. 21–31, Jul. 1996. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/092188909600022X

[4] O. O. Vergara Villegas, M. Nandayapa, and I. Soto, Eds., *Advanced Topics on Computer Vision, Control and Robotics in Mechatronics*. Cham: Springer International Publishing, 2018. [Online]. Available: http://link.springer.com/10.1007/978-3-319-77770-2

[5] J. Li, Z. Xu, D. Zhu, K. Dong, T. Yan, Z. Zeng, and S. X. Yang, "Bio-inspired intelligence with applications to robotics: a survey," *Intelligence & Robotics*, vol. 1, no. 1, pp. 58–83, Oct. 2021, publisher: OAE Publishing Inc. [Online]. Available: https://intellrobot.com/article/view/4353

[6] P. Tylkin, T.-H. Wang, K. Palko, R. Allen, H. C. Siu, D. Wrafter, T. Seyde, A. Amini, and D. Rus, "Interpretable autonomous flight via compact visualizable neural circuit policies," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3265–3272, 2022.

[7] N. S. Szczecinski and R. D. Quinn, "Template for the neural control of directed stepping generalized to all legs of MantisBot," *Bioinspiration & Biomimetics*, vol. 12, no. 4, p. 045001, Jun. 2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1748-3190/aa6dd9

[8] N. S. Szczecinski, A. J. Hunt, and R. D. Quinn, "A functional subnetwork approach to designing synthetic nervous systems that control legged robot locomotion," *Frontiers in neurorobotics*, vol. 11, p. 37, 2017.

[9] A. Hunt, N. Szczecinski, and R. Quinn, "Development and Training of a Neural Controller for Hind Leg Walking in a Dog Robot," *Frontiers in Neurorobotics*, vol. 11, 2017. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2017.00018

[10] N. S. Szczecinski, A. J. Hunt, and R. D. Quinn, "Design process and tools for dynamic neuromechanical models and robot controllers," *Biological cybernetics*, vol. 111, no. 1, pp. 105–127, 2017.

[11] S. Rubeo, N. Szczecinski, and R. Quinn, "A Synthetic Nervous System Controls a Simulated Cockroach," *Applied Sciences*, vol. 8, no. 1, p. 6, Jan. 2018, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2076-3417/8/1/6

[12] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons* . Oxford University Press, 11 1998. [Online]. Available: https://doi.org/10.1093/oso/9780195104912.001.0001

[13] V. A. Webster-Wood, J. P. Gill, P. J. Thomas, and H. J. Chiel, "Control for multifunctionality: bioinspired control based on feeding in aplysia californica," *Biological cybernetics*, vol. 114, no. 6, pp. 557–588, 2020.

[14] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Liquid time-constant networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7657–7666.

[15] K. ichi Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360800580125X

[16] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems*, vol. 31, 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf

[17] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014. [Online]. Available: http://arxiv.org/abs/1409.1259

[18] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré, "Combining recurrent, convolutional, and continuous-time models with linear state space layers," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 572–585. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/05546b0e38ab9175cd905eebcc6ebb76-Paper.pdf

[19] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[20] M. Lechner, R. Hasani, M. Zimmer, T. A. Henzinger, and R. Grosu, "Designing worm-inspired neural networks for interpretable robotic control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 87–94.

[21] M. Lechner, R. Hasani, A. Amini, T. A. Henzinger, D. Rus, and R. Grosu, "Neural circuit policies enabling auditable autonomy," *Nature Machine Intelligence*, vol. 2, no. 10, pp. 642–652, 2020.

[22] B. A. Bicknell and M. Häusser, "A synaptic learning rule for exploiting nonlinear dendritic computation," *Neuron*, vol. 109, no. 24, pp. 4001–4017, 2021.

[23] C. Koch and I. Segev, "The role of single neurons in information processing," *Nature neuroscience*, vol. 3, no. 11, pp. 1171–1177, 2000.

[24] L. N. Groschner, J. G. Malis, B. Zuidinga, and A. Borst, "A biophysical account of multiplication by a single neuron," *Nature*, vol. 603, no. 7899, pp. 119–123, 2022.

[25] S. M. Jayakumar, W. M. Czarnecki, J. Menick, J. Schwarz, J. W. Rae, S. Osindero, Y. W. Teh, T. Harley, and R. Pascanu, "Multiplicative interactions and where to find them," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=rylnK6VtDH

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[27] I. Hurwitz, I. Kupfermann, and K. R. Weiss, "Fast synaptic connections from CBIs to pattern-generating neurons in Aplysia: initiation and modification of motor programs," *Journal of Neurophysiology*, vol. 89, no. 4, pp. 2120–2136, 2003, pMID: 12686581. [Online]. Available: https://doi.org/10.1152/jn.00497.2002

[28] J. Jing and K. R. Weiss, "Neural mechanisms of motor program switching inaplysia," *Journal of Neuroscience*, vol. 21, no. 18, pp. 7349–7362, 2001. [Online]. Available: https://www.jneurosci.org/content/21/18/7349

[29] ——, "Generation of variants of a motor act in a modular and hierarchical motor network," *Current Biology*, vol. 15, no. 19, pp. 1712–1721, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960982205009863

[30] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[31] Official creality CR 10 s5 3d printer best large 3d printer. [Online]. Available: https://www.creality3dofficial.com/products/creality-cr-10-s5-3d-printer

[32] A. Stepanyants, P. R. Hof, and D. B. Chklovskii, "Geometry and Structural Plasticity of Synaptic Connectivity," *Neuron*, vol. 34, no. 2, pp. 275–288, Apr. 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0896627302006529

[33] D. W. Morton and H. J. Chiel, "Neural architectures for adaptive behavior," *Trends in Neurosciences*, vol. 17, no. 10, pp. 413–420, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0166223694900159