Article

CBMC: A Biomimetic Approach for Control of a 7-Degree of Freedom Robotic Arm

Qingkai Li [1], Yanbo Pang [1], Yushi Wang [1], Xinyu Han [1], Qing Li [1] and Mingguo Zhao [1,2,*]

[1] Department of Automation, Tsinghua University, Beijing 100084, China
[2] Beijing Innovation Center for Future Chips, Tsinghua University, Beijing 100084, China
[*] Correspondence: mgzhao@mail.tsinghua.edu.cn

Abstract: Many approaches inspired by brain science have been proposed for robotic control, specifically targeting situations where knowledge of the dynamic model is unavailable. This is crucial because dynamic model inaccuracies and variations can occur during the robot's operation. In this paper, inspired by the central nervous system (CNS), we present a CNS-based Biomimetic Motor Control (CBMC) approach consisting of four modules. The first module consists of a cerebellum-like spiking neural network that employs spiking timing-dependent plasticity to learn the dynamics mechanisms and adjust the synapses connecting the spiking neurons. The second module constructed using an artificial neural network, mimicking the regulation ability of the cerebral cortex to the cerebellum in the CNS, learns by reinforcement learning to supervise the cerebellum module with instructive input. The third and last modules are the cerebral sensory module and the spinal cord module, which deal with sensory input and provide modulation to torque commands, respectively. To validate our method, CBMC was applied to the trajectory tracking control of a 7-DoF robotic arm in simulation. Finally, experiments are conducted on the robotic arm using various payloads, and the results of these experiments clearly demonstrate the effectiveness of the proposed methodology.

Keywords: brain-inspired computing system; neuromorphic computing; spiking neural network; reinforcement learning; robotic arm

check for updates

Citation: Li, Q.; Pang, Y.; Wang Y.; Han X.; Li, Q.; Zhao, M. CBMC: A Biomimetic Approach for Control of a 7-Degree of Freedom Robotic Arm. *Biomimetics* **2023**, *8*, 389. https://doi.org/10.3390/biomimetics8050389

Academic Editor: Jinyou Shao

Received: 14 July 2023
Revised: 23 August 2023
Accepted: 24 August 2023
Published: 25 August 2023

1. Introduction

The past few years have seen a blossoming of robotic applications in various fields, including manufacturing, health care and customer service, etc. The key issue of developing robots up to these applications lies in the control ability of the manipulation. For a force-controlled robot, the mapping between joint torque commands and the end-effector position is often generated by a previously acquired dynamic model, whose accuracy plays a vital role in the control ability. However, due to the uncertainties of the working environment and the development of elastic, muscle-like actuators, the accurate modeling of a robot's dynamics is almost intractable in many scenarios. As a result of the evolution after billions of years, animals, especially human beings, have developed an adaptive control solution for motor performance that will work robustly in different environments with elastic muscles and joints, without the presence of a dynamic model, and can almost outperform the most state-of-art robots in many aspects. Therefore, researchers turn to bio-inspired approaches for inspiration.

Mimicking the learning ability of the cerebral cortex, some researchers have adopted artificial neural networks (ANNs), which are built by layers of computing neurons, as a solution for controlling robots without a dynamic model. In [1], an ANN-based control strategy is proposed for a flexible robotic arm with consideration of friction for both motor and payload. In the research of [2], an adaptive control method is introduced to the manipulator with unknown system dynamics. Wang et al. [3] take the output nonlinearity and unmodeled dynamics into consideration and develops an ANN module to approximate the unknown dynamics. In [4–6], some controlling methods based on ANN are also introduced to handle environment uncertainties and disturbance, like robotic manipulators working underwater. However, ANNs can only loosely model the functioning of the cerebral cortex. Artificial neurons in the network process information by nonlinear function of the sum of neuron inputs, and the output is transmitted through neuron connections and adjusted as the learning proceeds. These conventional neurons lack the ability to carry time-related information and thus the network can hardly deal with the temporal–spatial information of a robot during movement. Furthermore, the training process of ANNs, which is time and energy costing and computationally expensive, raises the stringency of demand on robotic processors [7].

To make up for the deficit, many other researchers have turn to spiking neural networks (SNNs) that mimic the underlying mechanisms of the brain more realistically. Unlike conventional neurons in ANNs, spiking neurons in SNNs precisely model the information transfer and processing as it happens in biological neurons, i.e., via discrete spikes that fire in certain timing patterns. This temporal coding mechanism in SNNs enables them to capture the temporal evolution of analog signals, making it a better solution for robotic control. Many works on SNN-based robotic applications have been presented. In [8,9], an SNN is trained with reinforcement learning to control a single-joint arm for target reaching. In [10,11], a 4-DoF robotic arm is controlled by a single-layer SNN network that is trained with spiking timing-dependent plasticity (STDP). Recently, DeWolf et al. [12] combined SNN and a neuromorphic chip to present a neurorobotic controller.

Despite the cerebral cortex, the involvement of the cerebellum in muscle and motor control has also been long advocated. Following this path, an SNN with cerebellum-inspired structure is presented in [13] for controlling a 2-DoF robotic arm, based on which a solution for compliant control and control under nondeterministic time delay is presented in [14,15], respectively.

The above studies have taken a positive step toward bio-inspired control in robots, mimicking some parts of the human brain in function or structure. However, their problem lies in viewing the cerebral cortex or the cerebellum as a stand-alone controller. An important observation about the brain is that schemas are distributed and computed in different brain areas. Motor control in vertebrates by the central nervous system (CNS) involves the cerebral motor cortex, basal ganglia, thalamus, cerebellum, brain stem, and spinal cord, and they work in collaboration in a hierarchical control loop [16]. It is therefore of great practical importance to study how this human control loop as a whole can be applied to robotic control.

In this paper, the main contributions are as follows:

- We propose a system model of the CNS-based Biomimetic Motor Control (CBMC) inspired by the human control loop for issues in control.
- A proposed implementation of this model involves utilizing an SNN for the cerebellum module, which is supervised by an ANN in the cerebral motor cortex module. This implementation is then applied to the control of a 7-DoF robotic arm.

The remainder of this paper is organized as follows: Section 2 presents the system model of CBMC. Section 3 will apply the above system to a 7-DoF robotic arm for demonstration. The results and discussion will be given in Section 4, and concluding remarks will be presented in Section 5.

2. CBMC: A Biomimetic Control Approach

In human motor control, several areas of the CNS, including the cerebral cortex, cerebellum, and spinal cord, contribute to the temporal–spatial coordination of the skeletomuscular system [17], as can be seen in Figure 1A [18]. The simplified control loop related to the cerebral cortex and cerebellum in supervising the spinal cord's control of the skeletomuscular system is depicted in Figure 1B [16]. Motor programs and commands are generated in the cerebral cortex, and the motor program is fed into the cerebellum, which sends out motor commands combining programs from the cerebral cortex and the sensory information from the spinal cord. The motor commands from the cerebral cortex and the cerebellum are then summed and sent out to the muscle via the brain stem and the spinal cord. The structure of the human motor control loop gives us some insights into how the CNS controls body movements.

Mimicking the CNS, the CBMC we proposed is shown in Figure 1C. It comprises four parts: the cerebral motor cortex module (CMCM), the cerebral sensory cortex module, the cerebellum module, and the spinal cord module. The spinal cord module carries signals between the arm and the brain and, at the same time, controls some reflexes without involving the brain. Sensory feedback signals from the spinal cord module are then processed in the cerebral sensory cortex module and fed into the CMCM and the cerebellum. The CMCM can choose the appropriate actions and plan the trajectory's shape to finish the general target. In contrast, the cerebellum module, which is supervised by the CMCM, provides corrections to compensate for errors from nonlinearities, delays, Coriolis, etc., and ensures the smoothness of movement. Motion trajectory is generated in the CMCM by a planner and fed into the agent, modeled by an ANN, and into the cerebellum module, which is in the structure of a cerebellar-like SNN. By taking the trajectory from the planner and the sensory feedback information from the cerebral sensory cortex module, the agent will provide a cerebral torque and instructive inputs to the cerebellum module. In the cerebellum module, the sensory feedback, planned trajectory, and instructive inputs are combined and analyzed for a cerebellum torque response, which is then added with the cerebral torque to form a joint torque command that will be sent down to the spinal cord module. Finally, after being processed in the spinal cord module, the joint torque command is conveyed toward the robot for manipulation.

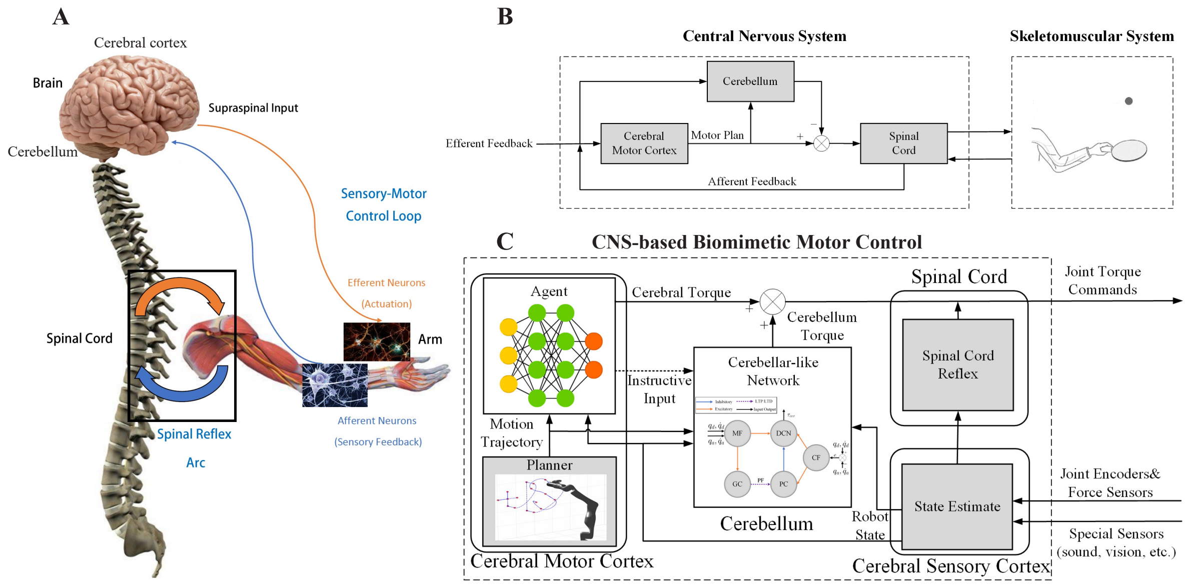

Figure 1. (**A**) Natural human motor control system. (**B**) Simplified human control loop related to the cerebral cortex, cerebellum, and spinal cord. (**C**) The CNS-based Biomimetic Motor Control (CBMC) control loop.

2.1. Cerebellum Module

To better demonstrate how the cerebellum module works, we will introduce it from three perspectives: the neuron model, the synaptic plasticity model, and the network structure.

2.1.1. Neuron Model

In an SNN, neural information transmitted between different neurons is carried in spike sequences, which can be defined as

$$S(t) = \sum_{f} \delta(t_f - t_0), \tag{1}$$

where $f = 1, 2, \ldots$ is the index label of a spike and $\delta(\cdot)$ is a Dirac function. The input signal $i(t)$ of a neuron from one synapse induced by a spike sequence can therefore be described as [19]

$$i(t) = \int_0^\infty S(s-t) exp(-s/\tau_s) ds, \tag{2}$$

where τ_s is a time constant.

In the existing literature, many spiking neural models have been proposed, such as the Hodgkin–Huxley [20] model as well as the Integrate-and-Fire model and its variants [21]. Although the Hodgkin–Huxley model has a better biomimetic reality, it is difficult to realize in the application for computation complexity. Maintaining the feature of membrane potential leakage in neurons and having a high computation efficiency, the Leaky-Integrate-and-Fire (LIF) model [22] is used as the spiking neuron model. The membrane potential of a LIF neuron u changes according to

$$\tau_m \frac{du}{dt}(t) = u_{reset} - u(t) + R(i_0(t) + \sum w_j i_j(t)) \tag{3}$$

where $\tau_m = RC$ is the time constant of the neuron membrane that models the voltage leakage. u_{reset} is the potential value after each reset. $i_0(t)$ stands for an external current driving the neural state, $i_j(t)$ denotes the input current from the j-th synapse, and w_j represents the strength of the j-th synapse. Once the membrane potential reaches the firing threshold u_{fire}, a single spike is fired from the neuron, and its potential is set back to u_{reset}.

2.1.2. Synaptic Plasticity Model

As seen in Equation (3), the neuron potential is influenced by the input synaptic weight w_j, which can be changed during the working process of the network. How to map the relationship between neuronal activity and the synaptic weights is what synaptic plasticity models will solve. Popular models can be classified into two types, namely, the rate-based and the spike-based. The latter has shown promising applications in robots and other autonomous systems [23], so we will take it here as an example.

The spike-based learning rule, often termed STDP, connects the weight change with the timing of individual spikes. If a presynaptic spike precedes a postsynaptic spike, then the synaptic activity will be strengthened, but if they happen in reversed order, then the synaptic activity will be weakened. The mathematical model of STDP can be given as [24]

$$\Delta w = \begin{cases} Ae^{\frac{-(|t_{pre} - t_{post}|)}{\tau_+}}, & t_{pre} - t_{post} \leq 0 \\ Be^{\frac{-(|t_{pre} - t_{post}|)}{\tau_-}}, & t_{pre} - t_{post} > 0 \end{cases} \tag{4}$$

where t_{pre} and t_{post} are the firing time of presynaptic neuron and postsynaptic neuron respectively, τ_+ ans τ_- are time constants, and $A > 0$ and $B < 0$ are constants scaling the change of weights, respectively.

2.1.3. Network Structure

Many computational models of the cerebellum have already been proposed, such as CMAC [25] and the Schweighofer–Arbib model [26]. The cerebellar-like network employed in our work is similar to that in [14,15], as depicted in Figure 2.

There are five different neural layers in this network: (1) mossy fibers (MFs); (2) granule cells (GCs); (3) climbing fibers (CFs); (4) Purkinje cells (PCs); (5) deep cerebellar nuclei (DCN). The desired and actual joint position and velocity are concatenated and coded into spiking patterns in the MF layer, which will then project excitatory afferent on both GC and DCN. The movement error of each joint will also be fed into CF and coded into spikes. GC will store and process the spiking pattern from MF and then generate spikes through parallel fibers (PFs) to PC. By combining the neural spike activity of both CF and GC, PC will accordingly give inhibitory afferent to DCN. Finally, joint torque commands will be produced by DCN combining spike information from MF, CF, and PC. The learning ability of this cerebellar-like network is achieved in the PFs by STDP.

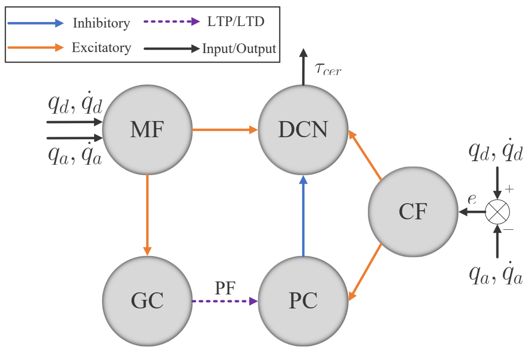

Figure 2. The cerebellar-like network.

2.2. *Cerebral Motor Cortex Module*

The CMCM is constructed of a feed-forward neural network, whose two main purposes are mimicking the dopamine mechanisms in baby learning and supervising the cerebellum module.

2.2.1. Learning Mechanism

The cerebral motor cortex plays an important role in human motion learning through trial and error, especially in babyhood. One of the complex learning mechanisms is induced by dopamine, which facilitates humans to replay newly acquired motions. The principle that humans learn from the consequences of their actions nowadays has been developed as the reinforcement learning (RL) method in artificial intelligence [27]. Therefore, RL is used to mimic the learning mechanism in CMCM, and the whole process can be modeled as a Markov decision process [28]:

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_1, \ldots, s_t). \tag{5}$$

The agent (cerebral motor cortex module) selects an action a_t at each time step with state s_t and policy π

$$a_t = \pi(s_t). \tag{6}$$

Then, the next state s_{t+1} is governed by a deterministic transition process

$$s_{t+1} = f(s_t, a_t) \tag{7}$$

and a reward r_{t+1} is returned from the state s_{t+1} with reward function

$$r_{t+1} = R(s_{t+1}). \tag{8}$$

Basically, the target of RL is to learn an optimal policy $\pi^\star$ at each time step to obtain a maximum cumulative reward

$$R_t = \sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k}, \tag{9}$$

where $\gamma \in [0, 1)$ because earlier rewards are more predictable than the long-term future reward, and the discount rate value helps avoid infinite returns in loopy Markov processes [29].

2.2.2. Supervision to the Cerebellum

As depicted in Figure 1C, the cerebellum module receives an instructive input from the CMCM, which influences the spiking firing rates of neurons in the cerebellum module and serves as supervision of the cerebellum to achieve a specific target. For instance, the CMCM will learn an additional movement to counteract the external force disturbances. A similar scene has been found in the cortex activities of monkeys when learning an arm-reaching task in a curl force field [30]. In addition, to demonstrate the different control levels between the CMCM and cerebellum module, a lower update frequency is employed in the CMCM.

3. Case Study: Trajectory Tracking Control of a 7-DoF Robotic Arm

3.1. Control Framework

In this section, we apply our method to trajectory tracking control tasks of a 7-DoF robotic arm, the Flexiv Rizon4s, as shown in Figure 3B. The whole control scheme is depicted in Figure 3A, where the direct output from the CMCM to the spinal cord module is omitted, and the dotted line implies a different frequency from other modules. In each control loop of the cerebellum module, the manipulator planner generates predefined reference position and orientation trajectories, and the joint trajectories are calculated with inverse kinematics. The cerebellum module receives the desired joint trajectories $\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d$ and feedback states $\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a$, then generates joint torques $\boldsymbol{\tau}_{cer}$. The spinal cord module here provides a gravity compensation torque. With every 20 loops of the cerebellum module running, the CMCM updates an additional zero-order-hold instructive input added to the desired joint trajectories based on the current targets and robot states.

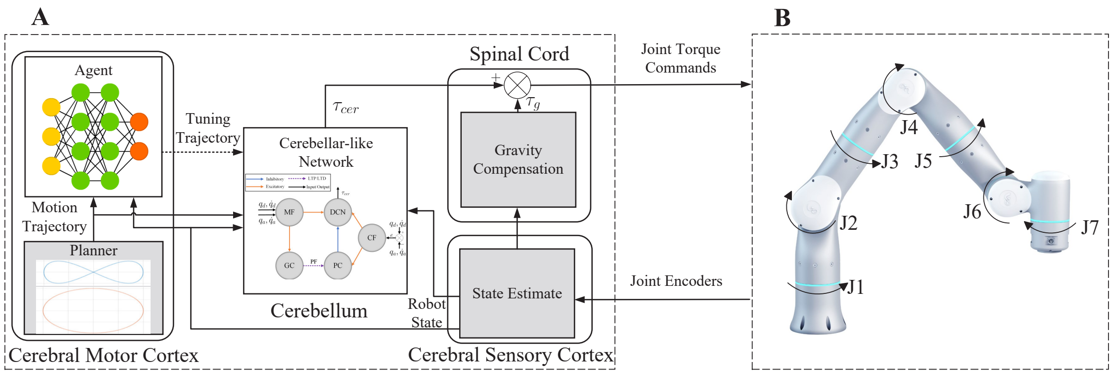

Figure 3. CBMC for the trajectory tracking of a 7-DoF robotic arm. (**A**) Overall control framework of CBMC. (**B**) The diagram of the 7-DoF robotic arm model.

3.2. Implementation of CBMC

3.2.1. Cerebellum-like SNN

In this paper, the cerebellum module is implemented with SpikingJelly [31], which is an open-source deep learning framework for SNN and has been used for exploring the applications of bio-inspired SNN in many aspects [32–34]. The cerebellum-like SNN described in Figure 2 consists of five layers: MFs, GCs, PCs, CFs, and DCN. All of them are divided into seven microcomplexes, each one for controlling a robot joint. In the MF–GC, CF–PC, CF–DCN, and PC–DCN connections, the seven microcomplexes are indeed independent, where the MF–GC, CF–PC, and CF–DCN connections act like encoders. However, the neurons from MFs to DCN and GCs to PCs are all fully connected, which means the seven microcomplexes are dependent. The MF–DCN connection generates a constant membrane voltage changing to both positive and negative torque neurons in DCN, which helps in reducing the noise influence. The GC–PC connection is where STDP learns the dynamic mechanism of the robot from the command information encoded in MF and the error information encoded in CF and improves the control effects of the cerebellum-like SNN. The following part will introduce how to implement the five layers in detail.

The MF layer has 40 spiking neurons per microcomplex, 280 in total, translating the analog information to spikes. For each joint, the 40 neurons are divided into four subgroups for encoding feedback and desired joint positions and velocities, respectively, with ten neurons each. For an analog value a with interval $[r_{\min}, r_{\max}]$, one spike $S_{i,\mathrm{MF}}(i = 1, 2, \ldots, 10)$ among the 10 neurons will be fired when

$$\begin{aligned} &a \in [c_{i-1}, c_i], \\ &c_k = r_{\min} + k \cdot \tfrac{(r_{\max} - r_{\min})}{10}, k = 0, 1, \ldots, 10. \end{aligned} \tag{10}$$

Therefore, 4 neurons per joint and 28 in total will be active at each time step. Every combination of four spikes is uniquely connected to one of 10,000 neurons per microcomplex in the GC layer with the excitatory synapse, represented by a positive weight $w_{\text{MF-GC}}$. All the neurons in the MF layer are concatenated together, fully connecting to the neurons in the DCN layer with excitatory synapse weight $w_{\text{MF-DCN}}$.

CF layer modifies the error between the desired and actual trajectories per joint to spikes with 100 spiking neurons per microcomplex. The front half of the 100 neurons are dedicated to the forward movement of each joint, and the back half are for joint reversing, which mimics the interaction between agonist and antagonist muscles in human movement. The normalized error value $e_j \in [-1, 1]$ of each joint is given as

$$e_j = \frac{q_{d,j} - q_{a,j} + \dot{q}_{d,j} - \dot{q}_{a,j}}{q_{\text{upper},j} - q_{\text{lower},j} + \dot{q}_{\text{upper},j} - \dot{q}_{\text{lower},j}}, \ j = 1, 2, \ldots, 7, \tag{11}$$

where $q_{d,j}, \dot{q}_{d,j}, q_{a,j}, \dot{q}_{a,j}$ are the desired and actual joint position and velocity, respectively, and $q_{\text{upper},j}, q_{\text{lower},j}, \dot{q}_{\text{upper},j}, \dot{q}_{\text{lower},j}$ are the upper and lower bounds of j-th joint position and velocity. Poisson encoding is applied depending on the error value of each joint to obtain the spikes $S_{j,i,\mathrm{CF}}$, which can be expressed as

$$S_{j,i,\mathrm{CF}} = \begin{cases} 1, & \text{if } |e_j| > \text{rand}(0,1) \\ 0, & \text{else.} \end{cases}, \ j = 1, 2, \ldots, 7. \tag{12}$$

In order to be consistent with the joint movement, only up to half of the neurons of the CF layer will be active per microcomplex, which means if $e_j > 0$, $i = 1, 2, \ldots, 50$; otherwise, $i = 51, 52, \ldots, 100$. Each neuron in the CF layer is connected one-to-one with each neuron in the PC layer and DCN layer with excitatory synapse weights $w_{\text{CF-PC}}$ and $w_{\text{CF-DCN}}$, respectively, also indicating the two other layers have the same number of neurons with the CF layer.

Neurons in the GC, PC, and DCN layers are all modeled as discrete-time LIF neurons to approximate the dynamics of the continuous-time LIF neurons. The membrane potential discrete-time charging function of the LIF neuron is

$$h[t] = v[t-1] - \frac{1}{\tau_m}(v[t-1] - v_{reset}) + x[t], \tag{13}$$

where τ_m is the voltage leaking time constant and $x[t]$ is the input from synapses. To avoid confusion, $h[t]$ is used to represent the membrane potential after neuronal charging but before neuronal firing at time t, $v[t]$ is the membrane potential after neuronal firing, and v_{reset} is the reset value of membrane potential. The reset function of the membrane potential $v[t]$ depending on the firing state is

$$v[t] = \begin{cases} v_{reset}, & \text{if } S[t] = 1 \\ h[t], & \text{else.} \end{cases} \tag{14}$$

The firing state of the LIF neuron is described as

$$S[t] = \begin{cases} 1, & \text{if } h[t] \geq v_{fire} \\ 0, & \text{else.} \end{cases}, \tag{15}$$

where v_{fire} is the firing threshold. Therefore, a LIF neuron will fire a spike when the membrane potential $h[t]$ reaches the firing threshold. All the configuration parameters of LIF neurons are summarized in Table 1.

Table 1. Model parameters of discrete-time LIF neurons.

Parameters	GC	PC	DCN
v_{reset}	0	0	0
v_{fire}	1.0	5.0	1.5
τ_m	50	60	12

The control mechanism of the cerebellum module is learned at the GC–PC connections by adjusting the synapses in PFs with the STDP mechanism. The trace method [35] is used to implement STDP and avoid recording all the firing times of presynaptic and postsynaptic neurons described in Equation (4). The update of synapse weight at time t with the trace method is

$$\Delta w_{i,j}[t] = f_{post}(w_{i,j}[t]) \cdot tr_i[t] \cdot S_j[t] - f_{pre}(w_{i,j}[t]) \cdot tr_j[t] \cdot S_i[t], \tag{16}$$

where indices i, j indicate the presynaptic and postsynaptic neurons, respectively, f_{post}, f_{pre} are functions constraining how weight changes, and $tr_i[t], tr_j[t]$ are the traces of the presynaptic and postsynaptic neurons that track their firing. The updated functions of the traces are

$$\begin{aligned} tr_i[t] &= tr_i[t-1] - \frac{tr_i[t-1]}{\tau_{pre}} + S_i[t] \\ tr_j[t] &= tr_j[t-1] - \frac{tr_j[t-1]}{\tau_{post}} + S_j[t], \end{aligned} \tag{17}$$

where τ_{pre}, τ_{post} are the time constants of the presynaptic and postsynaptic neurons, similar to the leakage of LIF neurons. $S_i[t], S_j[t]$ in both Equations (16) and (17) are the firing states of the presynaptic and postsynaptic neurons.

Receiving excitatory synapses from the GC and CF layers, the neurons in the PC layer are activated and then one-to-one connected to the neurons in the DCN layer but with an inhibitory synapse, represented by a negative weight $w_{\text{PC-DCN}}$. Table 2 summarizes all the synapse weights. Finally, combining all the excitatory synapses from MF and CF layers and inhibitory synapses from the CF layer, the neurons in the DCN layer generate spikes, and then those spikes are mapped to joint torques $\boldsymbol{\tau}_{cer}$. The decode function of each microcomplex is as follows

$$\tau_{cer,j} = \alpha_j \left(\sum_{i=1}^{50} S_{j,i}[t] - \sum_{i=51}^{100} S_{j,i}[t] \right), \tag{18}$$

where $j = 1, 2, \ldots, 7$ is corresponding to the joint number, α_j is the mapping factor transforming the spikes to torques and is set as $\boldsymbol{\alpha} = (4.5, 4.5, 4.5, 1.7, 2.7, 1.0, 0.05)\ \text{N} \cdot \text{m/spike}$.

Table 2. Parameters of synapse weights.

Synapses	$w_{\text{MF-GC}}$	$w_{\text{MF-DCN}}$	$w_{\text{CF-DCN}}$	$w_{\text{CF-PC}}$
value	0.25	0.0028	0.45	−0.5

3.2.2. CMCM with Deep Deterministic Policy Gradient

In this project, deep deterministic policy gradient (DDPG) [36] algorithm as the RL implementation in CMCM is adopted to supervise the cerebellum module, based on a deep reinforcement learning library PFRL [37]. DDPG is a model-free algorithm that learns the deterministic policy to the continuous action domain, as

$$\boldsymbol{a} = \mu(\boldsymbol{s}|\boldsymbol{\theta}^{\mu}) \tag{19}$$

where $\boldsymbol{a} \in \mathbb{R}^7$ is the action vector, $\boldsymbol{s} \in \mathbb{R}^{28}$ is the state vector, and $\boldsymbol{\theta}^{\mu}$ is the parameter of the policy network. The actions are interpreted as additional desired joint positions and added to the original position targets from the trajectory generator. The state vector $\boldsymbol{s}$ is spliced by the desired and actual joint positions and velocities.

In addition, an action-value function $Q(\boldsymbol{s}, \boldsymbol{a})$ is used in DDPG for describing the expected reward in Equation (9) after taking an action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$. Considering the function approximators parameterized by $\boldsymbol{\theta}^{Q}$, one target of the DDPG is minimizing the Bellman residual

$$L(\boldsymbol{\theta}^{Q}) = \mathbb{E}\left[\left(Q(\boldsymbol{s}_t, \boldsymbol{a}_t|\boldsymbol{\theta}^{Q}) - y_t\right)^2\right], \tag{20}$$

where

$$y_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma Q'(\boldsymbol{s}_{t+1}, \mu'(\boldsymbol{s}_{t+1}|\boldsymbol{\theta}^{\mu'})|\boldsymbol{\theta}^{Q'}). \tag{21}$$

Here, $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is the reward function and γ is the discount factor, Q', μ' are target networks. Another target of the DDPG is learning the policy, which is evaluated by maximizing the performance objective

$$J(\boldsymbol{\theta}^{\mu}) = \mathbb{E}[Q(\boldsymbol{s}_t, \mu(\boldsymbol{s}_t|\boldsymbol{\theta}^{\mu}))]. \tag{22}$$

For the trajectory-tracking tasks, the total reward f in one simulation step is a weighted sum of the punishment of the joint errors and Cartesian position error as

$$f = 0.5\sum_{j=1}^{7} f_{joint,j} + f_c, \tag{23}$$

where

$$\begin{aligned} f_{joint,j} &= \begin{cases} -0.1 & ,\text{if } \dot{q}_{des,j} \cdot \dot{q}_{act,j} < 0 \\ -10\|q_{des,j} - q_{act,j}\|^2 & ,\text{else.} \end{cases} \\ f_c &= -10\|\boldsymbol{x}_{des} - \boldsymbol{x}_{act}\|^2. \end{aligned} \tag{24}$$

A constant punishment is given if the desired and actual joint velocity direction are not the same. Otherwise, we punish the joint position errors. Here, $\boldsymbol{x}$ denotes the Cartesian position of the end-effector, and the punishment is set as the distance from the target to the estimated Cartesian position.

The whole training process is divided into two stages. First, the cerebellum module is pre-trained without the CMCM, then it is fixed, and the agent explores the tuning policy to the cerebellum module with the aid of the reward mechanism. The learning algorithm of CMCM is as shown in Algorithm 1.

We train our CBMC with a specific trajectory target, which is an inclined circle as described in Equation (25), and without payload on the end-effector in the PyBullet physics simulator [38], and 150 trials in each epoch. The initial state of the robot is not on the trajectory at the beginning. One hundred epochs, thus 15 k trials, are performed, and the learning curves of the actor network and critic network are shown in Figure 4. After this learning process, the controller is applied to different trajectory-tracking tasks and is faced with unknown payloads on the end-effector.

Algorithm 1 Learning algorithm of CBMC

```
1:  Load the cerebellum-like SNN
2:  Initialize main critic network, actor network, target networks, and replay buffer
3:  Initialize relative frequency $F$ between cerebellum module and CMCM
4:  for epoch $= 1$ to $N$ do
5:      Initialize the neurons states
6:      Initialize the robot states
7:      Initialize the period reward $r = 0$
8:      for $t = 1$ to $M$ do
9:          if ($t$ Mod $F$) == 1 then
10:             Generate action $\boldsymbol{a}_t$ according to current policy and states
11:             Update cerebellum-like SNN
12:         end if
13:         Calculate torque commands $\boldsymbol{\tau}$ from CBMC
14:         Execute torque commands and observe new states $\boldsymbol{s}_{t+1}$
15:         Compute reward $f$ and accumulate the period reward $r = r + f$
16:         if ($t$ Mod $F$) == 1 and $t \neq 1$ then
17:             Store transition $(\boldsymbol{s}_{t-F}, \boldsymbol{a}_{t-F}, R, \boldsymbol{s}_t)$ in replay buffer
18:             Reset the period reward $r = 0$
19:             Update the critic and actor networks by training on a small batch of samples
                from the replay buffer
20:             Update the target networks
21:         end if
22:     end for
23: end for
```

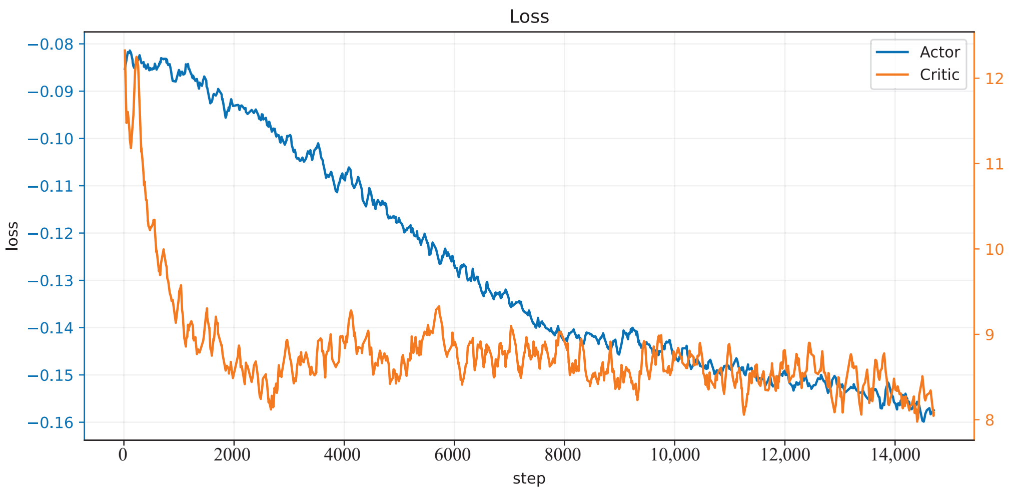

Figure 4. The loss curves of the actor and critic networks.

3.3. Experiment Settings

To assess the efficacy of our novel control strategy in robot dynamic control and trajectory tracking, we execute experiments considering two key factors. On the one hand, we test our CBMC on the robotic arm with different payloads on the end-effector in smooth trajectories. A single-joint movement will cause interaction forces to all other joints. The disturbance force cannot be compensated easily on the condition that the dynamics model is unknown. On the other hand, we test our CBMC controlling the end-effector tracking different trajectories containing a circle trajectory in the inclined plane and an eight-like trajectory in the horizontal plane, covering most of the possible translation motions of the robotic arm in the Cartesian space. The circular and eight-like trajectories are described in Equation (25).

$$\text{Circle:}\begin{cases} x = x_0 + R_c \cdot \cos(2\pi t/T_c) \cdot \cos\theta \\ y = y_0 + R_c \cdot \sin(2\pi t/T_c) \\ z = z_0 + R_c \cdot \cos(2\pi t/T_c) \cdot \sin\theta \end{cases} \quad \text{Eight-like:}\begin{cases} x = x_0 + R_e \cdot \cos(2\pi t/T_e) \\ y = y_0 + 0.5R_e \cdot \sin(4\pi t/T_e)\,, \\ z = z_0 \end{cases} \tag{25}$$

where $R_c = R_e = 0.14\,\mathrm{m}$ is the radius of the trajectory, $T_c = T_e = 3\,\mathrm{s}$ is the period, and $\theta = 30^\circ$ is the slant angle of the circle along the horizontal plane. (x_0, y_0, z_0) is used to adapt the trajectory within the workspace of robot. Providing the 3-D position and maintaining the orientation of the end-effector, the joint trajectories are calculated through an offline process using the inverse kinematics of the robot.

The performance of the CBMC on trajectory tracking is evaluated by comparing the desired and the actual joint positions. We use the mean square error (MSE) as the metric to evaluate the errors described in the following equations:

$$\begin{aligned} \mathrm{MSE}_j &= \frac{1}{K}\sum_{t=1}^{K} \|q_{d,j}[t] - q_{a,j}[t]\|^2 \\ \mathrm{MSE} &= \frac{1}{N}\sum_{j=1}^{N} \mathrm{MSE}_j, \end{aligned} \tag{26}$$

where $K = 3 \times 10^4$ denotes the simulation timestep number, corresponding to 10 cycles of the trajectories, and $N = 7$ is the number of joints.

4. Results and Discussion

In this section, we outline the experimental results that show how our method works and verify its effectiveness in trajectory tracking facing unknown payloads. To demonstrate that, the performances of the control effects with payloads of 0, 0.5, and 2.5 kg are studied on the aforementioned inclined circle and eight-like trajectories. On the other hand, we also evaluate our controller in target reaching task, whose movement is an s-curve toward a target point over time, to show the ability to face irregular but usual movements in human daily life.

Firstly, a brief description of the neuron activities in the control process is described in Figure 5, which shows the DCN neurons' activities in the first three cycles under the condition of inclined circle trajectory and no payload on the end-effector. There are seven hundred neurons, and each hundred corresponds to a joint actuator. When the membrane voltage of a neuron reaches its firing threshold, which is set as 1.5 in Figure 5A, one can see a corresponding spike is fired in Figure 5B.

Taking the first joint as an example, the first fifty neuron spikes will generate a positive acceleration and, therefore, dense spikes are fired in the beginning time as shown in Figure 5B to accelerate the robotic arm from a static state to the desired trajectory. In contrast, the last fifty neuron spikes will generate a deceleration by negative torque. Thus, combined with the orange dotted line in Figure 6, which is the corresponding joint tracking trajectory, indicating deceleration of the first joint around 800, 3800 ms and acceleration around 2200, 5200 ms, we find it is consistent with the DCN neuron spikes' activity as shown in Figure 5B, where dense spikes of the last fifty neurons are fired when there is deceleration and the first fifty are fired when there is acceleration. This phenomenon is not obvious in the last three joints because of the small torques for the same joint movements.

Different payloads, as mentioned before, are tested on the inclined circle trajectory. Figure 6 shows the joint trajectories tracking curves of the CBMC with different payloads in the first three cycles, and Figure 7A shows end-effector tracking curves in the whole ten cycles. The results show the reliable ability of the CBMC when the dynamics of the robotic arm change.

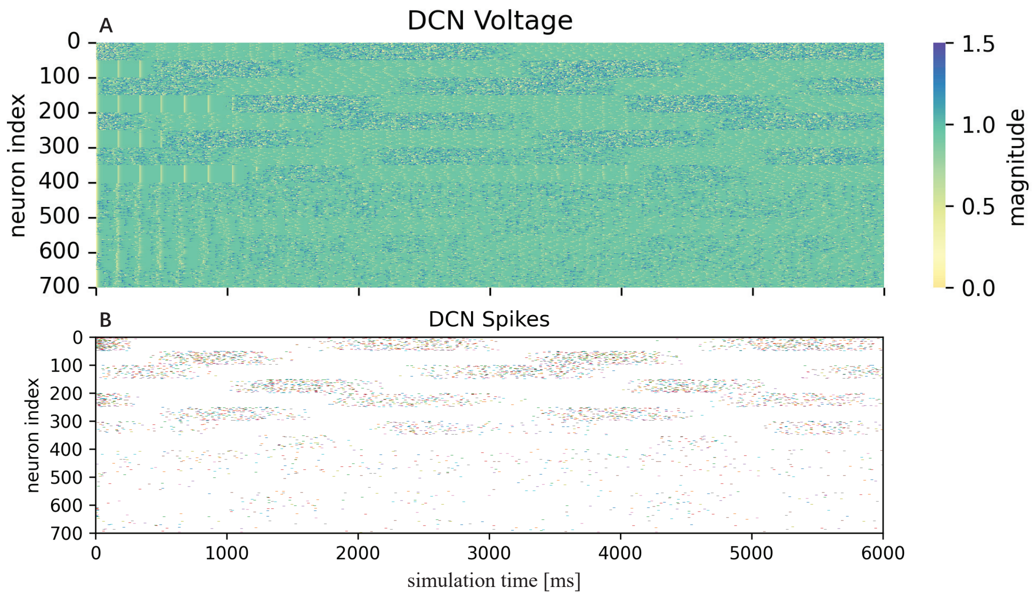

Figure 5. The DCN neurons' activities in the first three cycles under the condition of inclined circle trajectory and no payload. (**A**) The heat map of the membrane voltages of the DCN neurons. (**B**) The corresponding spike's firing states.

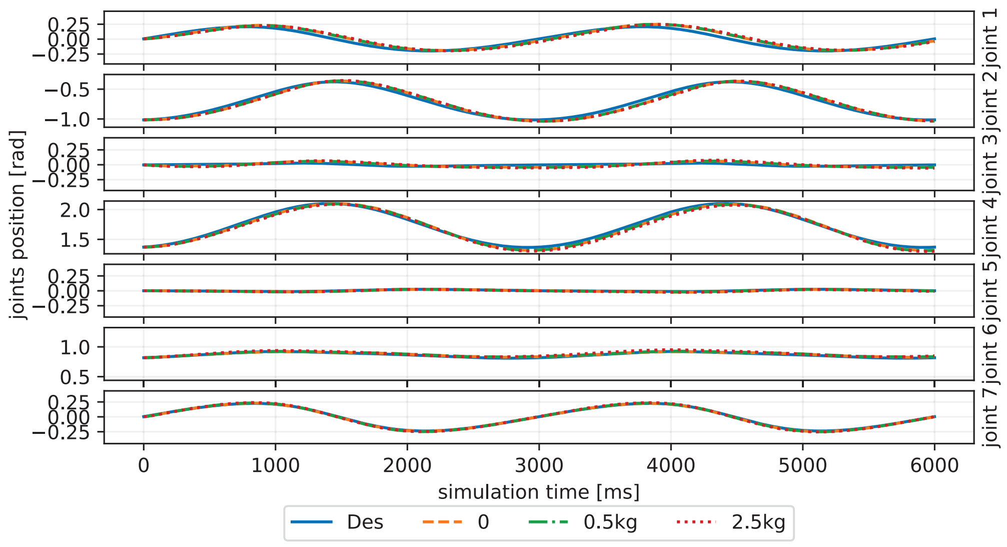

Figure 6. The joint trajectories tracking curves of the CBMC to different payloads in the first three cycles under the inclined circle trajectory condition.

In addition, we also test the CBMC on the eight-like trajectory that it never learns in the training process. Compared to the circle trajectory, the eight-like trajectory requires a faster and steeper change of the velocity and direction in the Cartesian space, resulting in more joint disturbances. Nonetheless, as depicted in the Cartesian trajectories of Figure 7B and the joint trajectories of Figure 8, the CBMC still shows a good performance on the condition of unknown trajectory and payloads. Table 3 lists the MSE of the joint position error under different trajectories and payloads. The CBMC shows a similar MSE loss in the Eight-like trajectory of about 1.2×10^{-4} compared to the training inclined circle trajectory for all conditions of payloads.

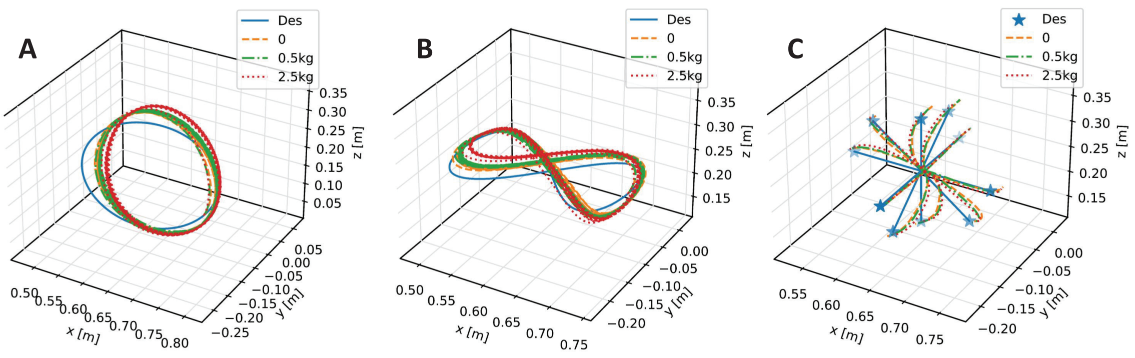

Figure 7. The desired versus actual trajectories of the end-effector with different payloads in the Cartesian space. (**A**) The inclined circle trajectory in Cartesian space. (**B**) The 8-like trajectory in Cartesian space. (**C**) The target reaching task trajectory in Cartesian space.

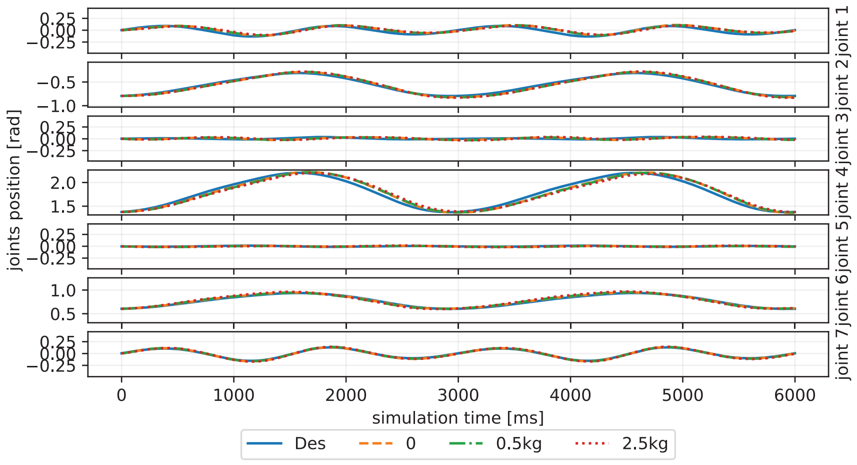

Figure 8. The joint trajectories tracking curves of the CBMC to different payloads in the 8-like trajectory condition.

Table 3. MSE of the joint position error under different trajectories and payloads.

Trajectory	**No Payload** $(\times 10^{-4})$	**0.5 kg** $(\times 10^{-4})$	**2.5 kg** $(\times 10^{-4})$
Inclined Circle	7.134 ± 0.749	8.250 ± 1.075	12.825 ± 1.795
Eight-Like Trajectory	8.340 ± 0.725	9.642 ± 0.819	13.862 ± 0.934
Target Reaching	1.266 ± 0.646	1.309 ± 0.711	1.633 ± 0.936

The target-reaching task consists of ten different reaching targets as the star markers shown in Figure 7C, which are around the same starting point. The challenges are interaction forces caused by acceleration and deceleration at the beginning and end, and the irregular directions toward different targets. Nevertheless, the CBMC performs a capable result in the target-reaching task from the Cartesian trajectory as shown in Figure 7C and the MSE in Table 3.

Finally, to demonstrate the effectiveness of the CBMC in dealing with unknown dynamics changes, we compare it with a PD controller on joint space. The PD controller is designed to have a similar (and even a little better) performance with CBMC on the inclined circle trajectory-tracking task and with no payload, the condition in which the CBMC is trained. Table 4 lists the MSE of different methods on inclined circle and with different payloads, where the method "No CMCM" corresponds to our method but removing the instructive input from CMCM. When the payload is added to the end-effector, we can see the PD controller performs worse than the CBMC. Particularly for the payload of 2.5 kg, the

MSE of PD increases about 52% compared to the CBMC. Combining the MSE of the "No CMCM", we can conclude that both the instructive inputs from CMCM and the dynamics mechanism learned in cerebellum module cause the CBMC have a better performance on this problem. In addition, the comparison between the CBMC and "No CMCM" implies the contribution of the supervising mechanism from the CMCM to the cerebellum module.

Table 4. MSE of different payloads in inclined circle: comparing to different methods.

Methods	**No Payload** $(\times 10^{-4})$	**0.5 kg** $(\times 10^{-4})$	**2.5 kg** $(\times 10^{-4})$
No CMCM	7.845 ± 0.898	8.883 ± 1.021	14.064 ± 1.805
CBMC	7.134 ± 0.749	$\mathbf{8.250 \pm 1.075}$	$\mathbf{12.825 \pm 1.795}$
PD	$\mathbf{7.036 \pm 0.219}$ [1]	8.740 ± 0.356	19.496 ± 1.989

[1] The bold number implies the minimum MSE under the corresponding payload condition.

5. Conclusions and Future Work

In this paper, inspired by the human control loop that outlines the CNS, we propose the CBMC approach, which mainly consists of four parts: the cerebral motor cortex module, the cerebellum module, the cerebral sensory cortex module, and the spinal cord module. Mimicking the biological feature in the human motor control system, the cerebellum module constructed by SNN aims to learn the dynamics feature of the robot, and it is supervised by instructive inputs from the cerebral motor cortex module, which learns using RL. The cerebral sensory cortex module deals with feedback information, including self-perception and environment interaction, while the spinal cord module modulates torque commands.

The proposed method was applied to controlling a 7-DOF robotic arm and partially simplified in the trajectory-tracking task, where the DDPG was used as the RL algorithm in the cerebral motor cortex module and the cerebellum-like SNN was implemented in the cerebellum module. To validate its effectiveness, we firstly trained the CBMC in a specific inclined circle trajectory-tracking task with no payload on the end-effector, then we verified its performances on the condition of different payloads and a new eight-like trajectory. Finally, we compared it to a PD controller to demonstrate the effectiveness of the supervising mechanism and the cerebellum-like SNN.

One limitation of this work is that the method is only validated in the simulation because the spiking neuron model with Python is not feasible for temporary torque control in the real robot manipulator. In the future, we will develop the proposed approach on real robotic arms. In addition, the ability of the CMCM can be explored in more complex tasks combining the cerebral sensory cortex module, like interacting with the environment, and the spinal cord module can be considered to control the rhythmic motion as a part of the whole motion control system.

Author Contributions: Conceptualization, Q.L. (Qingkai Li) and Y.P.; methodology, Q.L. (Qingkai Li) and X.H.; software, Q.L. (Qingkai Li) and Y.P.; validation, Q.L. (Qingkai Li), Y.P. and Y.W.; formal analysis, Q.L. (Qingkai Li); investigation, Q.L. (Qingkai Li), Y.P. and Y.W.; resources, M.Z.; data curation, Q.L. (Qingkai Li); writing—original draft preparation, Q.L. (Qingkai Li) and Y.P.; writing—review and editing, Q.L. (Qingkai Li), Y.P., X.H. and Y.W.; visualization, Q.L. (Qingkai Li) and Y.W.; supervision, Q.L. (Qing Li); project administration, M.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by STI 2030—Major Projects grant number 2021ZD0201402.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DoF	Degree of Freedom
ANN	artificial neuron network
CNS	central nervous system
CBMC	CNS-based Biomimetic Motor Control
SNN	spiking neural network
RL	reinforcement learning
STDP	spiking timing-dependent plasticity
CMCM	cerebral motor cortex module
LIF	Leaky-Integrate-and-Fire
MF	mossy fiber
GC	granule cell
CF	climbing fiber
PC	Purkinje cell
DCN	deep cerebellar nuclei
PF	parallel fiber

References

1. Chaoui, H.; Sicard, P.; Gueaieb, W. ANN-Based Adaptive Control of Robotic Manipulators With Friction and Joint Elasticity. *IEEE Trans. Ind. Electron.* **2009**, *56*, 3174–3187.
2. He, W.; Dong, Y. Adaptive Fuzzy Neural Network Control for a Constrained Robot Using Impedance Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 1174–1186.
3. Wang, F.; Liu, Z.; Chen, C.L.P.; Zhang, Y. Adaptive neural network-based visual servoing control for manipulator with unknown output nonlinearities. *Inf. Sci.* **2018**, *451–452*, 16–33.
4. Salloom, T.; Yu, X.; He, W.; Kaynak, O. Adaptive Neural Network Control of Underwater Robotic Manipulators Tuned by a Genetic Algorithm. *J. Intell. Robot. Syst.* **2020**, *97*, 657–672.
5. Liu, C.; Zhao, Z.; Wen, G. Adaptive neural network control with optimal number of hidden nodes for trajectory tracking of robot manipulators. *Neurocomputing* **2019**, *350*, 136–145.
6. Pham, D.T.; Nguyen, T.V.; Le, H.X.; Nguyen, L.V.; Thai, N.H.; Phan, T.A.; Pham, H.T.; Duong, A.H. Adaptive neural network based dynamic surface control for uncertain dual arm robots. *Int. J. Dyn. Control* **2019**, *8*, 824–834.
7. Liu, Z.; Peng, K.; Han, L.; Guan, S. Modeling and Control of Robotic Manipulators Based on Artificial Neural Networks: A Review. *Iran. J. Sci. Technol. Trans. Mech. Eng.* **2023**, 1–41, https://doi.org/10.1007/s40997-023-00596-3.
8. Chadderdon, G.L.; Neymotin, S.A.; Kerr, C.C.; Lytton, W.W. Correction: Reinforcement Learning of Targeted Movement in a Spiking Neuronal Model of Motor Cortex. *PLoS ONE* **2013**, *8*, e47251.
9. Spüler, M.; Nagel, S.; Rosenstiel, W. A spiking neuronal model learning a motor control task by reinforcement learning and structural synaptic plasticity. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–8.
10. Bouganis, A.; Shanahan, M. Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity. In Proceedings of the The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–8.
11. Chen, X.; Zhu, W.; Dai, Y.; Ren, Q. A bio-inspired spiking neural network for control of a 4-dof robotic arm. In Proceedings of the 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), Kristiansand, Norway, 9–13 November 2020; pp. 616–621.
12. DeWolf, T.; Patel, K.; Jaworski, P.; Leontie, R.; Hays, J.; Eliasmith, C. Neuromorphic control of a simulated 7-DOF arm using Loihi. *Neuromorphic Comput. Eng.* **2023**, *3*, 014007.
13. Carrillo, R.R.; Ros, E.; Boucheny, C.; Olivier, J.M.C. A real-time spiking cerebellum model for learning robot control. *Biosystems* **2008**, *94*, 18–27.
14. Abadia, I.; Naveros, F.; Garrido, J.A.; Ros, E.; Luque, N.R. On robot compliance: A cerebellar control approach. *IEEE Trans. Cybern.* **2019**, *51*, 2476–2489.
15. Abadía, I.; Naveros, F.; Ros, E.; Carrillo, R.R.; Luque, N.R. A cerebellar-based solution to the nondeterministic time delay problem in robotic control. *Sci. Robot.* **2021**, *6*, eabf2756.
16. Van Der Smagt, P.; Arbib, M.A.; Metta, G. Neurorobotics: From vision to action. In *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 2069–2094.
17. Swinnen, S.P.; Vangheluwe, S.; Wagemans, J.; Coxon, J.P.; Goble, D.J.; Van Impe, A.; Sunaert, S.; Peeters, R.; Wenderoth, N. Shared neural resources between left and right interlimb coordination skills: The neural substrate of abstract motor representations. *Neuroimage* **2010**, *49*, 2570–2580.
18. Bhat, A. *A Soft and Bio-Inspired Prosthesis with Tactile Feedback*; Carnegie Mellon University: Pittsburgh, PA, USA, 2017.

19. Ponulak, F.; Kasinski, A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta Neurobiol. Exp.* **2011**, *71*, 409–433.
20. Hodgkin, A.L.; Huxley, A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **1952**, *117*, 500.
21. Burkitt, A.N. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. *Biol. Cybern.* **2006**, *95*, 1–19.
22. Stein, R.B. A theoretical analysis of neuronal variability. *Biophys. J.* **1965**, *5*, 173–194.
23. Bing, Z.; Meschede, C.; Röhrbein, F.; Huang, K.; Knoll, A.C. A survey of robotics control based on learning-inspired spiking neural networks. *Front. Neurorobot.* **2018**, *12*, 35.
24. Gerstner, W.; Kistler, W.M. Mathematical formulations of Hebbian learning. *Biol. Cybern.* **2002**, *87*, 404–415.
25. Albus, J.S. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *J. Dyn. Sys. Meas. Control* **1975**, *97*, 220–227.
26. Schweighofer, N. *Computational Models of the Cerebellum in the Adaptive Control of Movements*; University of Southern California: Los Angeles, CA, USA, 1995.
27. Holroyd, C.B.; Coles, M.G.H. The neural basis of human error processing: Reinforcement learning, dopamine, and the error-related negativity. *Psychol. Rev.* **2002**, *109*, 679–709.
28. Sutton, R.S.; Barto, A.G. Reinforcement Learning: An Introduction. *IEEE Trans. Neural Netw.* **2005**, *16*, 285–286.
29. Nagaraj, A.; Sood, M.; Patil, B.M. A Concise Introduction to Reinforcement Learning in Robotics. *arXiv* **2022**, arXiv:2210.07397.
30. Sun, X.; O'Shea, D.J.; Golub, M.D.; Trautmann, E.M.; Vyas, S.; Ryu, S.I.; Shenoy, K.V. Cortical preparatory activity indexes learned motor memories. *Nature* **2022**, *602*, 274–279.
31. Fang, W.; Chen, Y.; Ding, J.; Chen, D.; Yu, Z.; Zhou, H.; Timothée M.; Tian, Y. SpikingJelly. 2020. Available online: https://github.com/fangwei123456/spikingjelly (accessed on 18 April 2023).
32. Yuan, R.; Duan, Q.; Tiw, P.J.; Li, G.; Xiao, Z.; Jing, Z.; Yang, K.; Liu, C.; Ge, C.; Huang, R.; et al. A calibratable sensory neuron based on epitaxial VO2 for spike-based neuromorphic multisensory system. *Nat. Commun.* **2022**, *13*, 3973.
33. Xiang, S.; Zhang, T.; Jiang, S.; Han, Y.; Zhang, Y.; Du, C.; Guo, X.; Yu, L.; chun Shi, Y.; Hao, Y. Spiking SiamFC++: Deep Spiking Neural Network for Object Tracking. *arXiv* **2022**, arXiv:2209.12010.
34. Liu, G.; Deng, W.; Xie, X.; Huang, L.; Tang, H. Human-Level Control through Directly-Trained Deep Spiking Q-Networks. *IEEE Trans. Cybern.* **2022**, 1–12, doi: 10.1109/TCYB.2022.3198259 .
35. Morrison, A.; Diesmann, M.; Gerstner, W. Phenomenological models of synaptic plasticity based on spike timing. *Biol. Cybern.* **2008**, *98*, 459–478.
36. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.M.O.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
37. Fujita, Y.; Nagarajan, P.; Kataoka, T.; Ishikawa, T. ChainerRL: A Deep Reinforcement Learning Library. *J. Mach. Learn. Res.* **2021**, *22*, 3557–3570.
38. Coumans, E.; Bai, Y. PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning. 2016–2021. Available online: http://pybullet.org (accessed on 21 May 2022).