Neuromorphic Computing and Engineering

PAPER • **OPEN ACCESS**

A cloud-edge framework for energy-efficient event-driven control: an integration of online supervised learning, spiking neural networks and local plasticity rules

To cite this article: Reza Ahmadvand *et al* 2024 *Neuromorph. Comput. Eng.* **4** 044004

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
10 March 2024

REVISED
15 October 2024

ACCEPTED FOR PUBLICATION
29 October 2024

PUBLISHED
7 November 2024

PAPER

A cloud-edge framework for energy-efficient event-driven control: an integration of online supervised learning, spiking neural networks and local plasticity rules

Reza Ahmadvand, Sarah Safura Sharif and Yaser Mike Banad*

School of Electrical and Computer Engineering, University of Oklahoma, Oklahoma, United States of America
* Author to whom any correspondence should be addressed.

E-mail: bana@ou.edu, iamrezaahmadvand1@ou.edu and s.sh@ou.edu

Keywords: spiking neural network, cloud-edge framework, obstacle avoidance, supervised learning, local plasticity

Supplementary material for this article is available online

Abstract

This paper presents a novel cloud-edge framework for addressing energy constraints in complex control systems. Our approach centers around a learning-based controller using Spiking Neural Networks (SNN) on physical plants. By integrating a biologically plausible learning method with local plasticity rules, we harness the energy efficiency, scalability of the newtwork, and low latency of SNNs. This design replicates control signals from a cloud-based controller directly on the plant, reducing the need for constant plant-cloud communication. The plant updates weights only when errors surpass predefined thresholds, ensuring efficiency and robustness in various conditions. Applied to linear workbench systems and satellite rendezvous scenarios, including obstacle avoidance, our architecture dramatically lowers normalized tracking error by 96% with increased network size. The event-driven nature of SNNs minimizes energy consumption, utilizing only about 11.1 $\times$ 104 pJ (0.3% of conventional computing requirements). The results demonstrate the system's adjustment to changing work environments and its efficient use of energy resources, with a moderate increase in energy consumption of 37% for dynamic obstacles, compared to non-obstacle scenarios.

1. Introduction

The pursuit of autonomy in next-generation robotic systems faces significant challenges related to task complexity, energy efficiency, and the latency of control frameworks. Advancements in brain-inspired computing, such as neuromorphic systems [1], offer promising solutions to these challenges. However, for dynamical systems leveraging centralized control methods, the limitations of traditional approaches, particularly in terms of energy efficiency and latency, restrict the implementation of complex control systems. Cloud-based control systems [2, 3] have been proposed to mitigate these limitations, but they introduce new challenges, such as the need for continuous communication between the physical plant and the cloud, leading to potential signal access issues and high energy consumption for data transfer. This necessitates the development of energy-efficient and fast learning-based controllers that can operate directly on the plant.

Spiking neural networks (SNNs) emerge as an energy-efficient alternative with low latency for complex data processing and decision-making on the plant [4]. However, the effectiveness of these networks largely depends on their capability to continuously adapt their learnable parameters. Particularly, for on-chip implementation, these learning rules must adhere to three principles: (1) adopting event-based computation through discrete spikes to guarantee superior energy efficiency; (2) facilitating local parallel information processing for cost-effective, near-memory, and asynchronous computation; and (3) enabling low-latency inferencing in neurons without the need for dependency on future data [5]. Local learning rules, the focus of this study, rely solely on locally available information for synaptic weight updates [5–9].

The application of these local learning rules in SNNs has been instrumental in harnessing the benefits of neuromorphic structures within robotic systems, leading to enhanced energy efficiency, scalability of the network, and robustness against neuron silencing. Leveraging supervised local learning rules, SNNs have been implemented for various applications [1, 10], including adaptive control for robotic arms [11, 12]. Despite their potential, existing local learning rules often fail to simultaneously address the critical factors of locality, robustness, and spiking efficiency—key aspects for practical deployment in neuromorphic hardware. To address these challenges, Alemi *et al* [13] proposed a supervised local learning rule that integrates the principles of efficient-balanced networks (EBN) [14] with nonlinear adaptive control theories, considering the nonlinear dendritic computations. Their approach leverages basis functions from adaptive control theories to accommodate nonlinear dendritic computations, enabling the learning of basis function coefficients by considering the reductions in the error signal. Consequently, the proposed learning rule not only benefits from the spiking efficiency and robustness inherent in EBNs but also incorporates the locality advantages offered by adaptive control theories.

Building upon this perspective and considering recent advancements in neuromorphic computing applications, such as the SNN-based framework for concurrent estimation and control in linear dynamical systems [15], we aim to address the limitations of these frameworks by applying them to satellite rendezvous maneuvers. This approach, an extension of the method by Slijkhuis *et al* that combines a Luenberger observer and a linear quadratic regulator (LQR) controller [16], introduces the SNN-based Modified Sliding Innovation Filter (SNN-MSIF) as a robust neuromorphic estimator for linear systems [17], leveraging the gain formulation introduced in [18]. Although these frameworks exhibit commendable accuracy and robustness in the presence of modeling uncertainties, neuron silencing, and measurement outliers, the absence of a learning mechanism hinders their adaptability and limits their potential for real-time applications.

Our study aims to harness the advantages of the developments in [13]. Drawing inspiration from supervised training methods applied to artificial neural network (ANN) for control systems such as the cartpole [19], we propose a cloud-edge-based, online supervised training framework by employing a recurrent EBN composed of leaky integrate-and-fire (LIF) neurons. The key feature of our SNN is its ability to quickly learn the control policy, requiring only a limited number of epochs to replicate the control signal generated by a controller operating on the cloud. A notable aspect of our approach is that the implemented SNN is designed to be independent of the underlying dynamics of the system and the controller. As such, none of the SNN parameters is required to be trained about the adopted controller (source of the desired signal) or dynamical plant (which receives the control signal reproduced by SNN at the edge). This independence makes our method highly versatile and applicable to a wide range of systems, regardless of the complexity of their dynamics. While alternative training methods exist [11, 20], our approach leverages the novel supervised local learning rule introduced in [13], which combines the benefits of EBNs with nonlinear adaptive control theories. This unique combination guarantees the spiking efficiency as well as the locality and robustness which is crucial particularly in the context of complex, real-world applications.

To assess the efficacy of our approach, we first applied it to a linear workbench problem in which the SNN successfully learned to reproduce the control signal from an LQR [21]. The SNN then used this learned signal to control a desired plant by reproducing this signal. Building on this, we extended our approach to a more complex and realistic scenario: satellite rendezvous with obstacle avoidance maneuver, considering the dynamic obstacles crucial for real-world problems in space robotics [22] in the presence of uncontrolled space objects/debris [23]. For this application, the model-based LQR controller on the cloud was designed using the Clohessy–Wiltshire (CW) equations [24]. Considering the results of [25] on the comparsion between SNNs and ANNs, and the efficiency of neuromorphic computers in practice, Our findings revealed that the proposed framework by this research offers a new framework with an acceptable performance in terms of accuracy and energy efficiency the traditional von Neumann-based architectures. Also, we have shown that the implemented SNN adjusts its spiking activity with changes in working conditions, and we have demonstrated that the level of energy consumption of SNN-based frameworks increases with the complexity of the considered task [26].

This paper is organized as follows: section 2 provides an overview of related works and their contributions to neuromorphic control and estimation in robotic systems. Then, section 2 delves into the theoretical foundations of our proposed approach, outlining the preliminaries, underlying theories, and the proposed framework for addressing the problem of concurrent robust estimation and control in linear dynamical systems. Section 3 presents the results of numerical simulations, showcasing the performance and effectiveness of our proposed framework and offering insightful discussions on the obtained results. Section 4 presents numerical simulations and discussions of the results obtained from these simulations. Finally, section 4 concludes the paper by summarizing key findings, discussing the implications of our research, and highlighting potential avenues for future work.

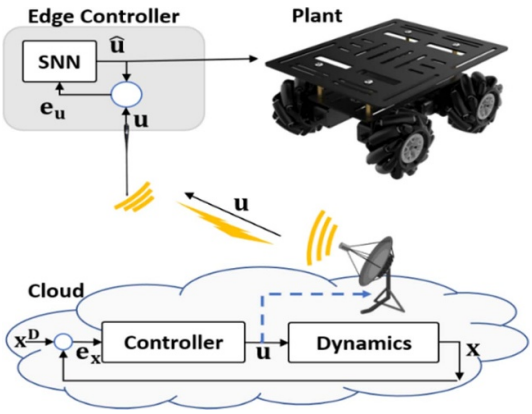

Figure 1. A schematic representation of a cloud-edge framework employing online supervised learning. The 'Rover' serves as an illustrative example for plant within an edge environment.

2. Theory

This section elucidates the theoretical underpinnings and outcomes of our study, focusing on the application of SNNs in centralized control systems. Our approach leverages cloud-based computing to address computational complexities in centralized control systems [2, 3]. In this cloud-centric architecture, control signals are computed in the cloud and transmitted to the physical plant via various antennae. While this setup necessitates continuous plant-cloud communication, it also poses a risk of losing control in case of communication failures or interruptions. To address this challenge, our study introduces an innovative architecture, as presented in figure 1. This architecture features an SNN-based onboard controller on the plant, capable of online training to reproduce control signals generated in the cloud. This design significantly relaxes the need for continuous plant-cloud communication, thereby enhancing the system's robustness in signal access denied conditions.

Moreover, our proposed architecture in this study leverages the advantages of neuromorphic computing such as energy efficiency and network scalability inherent in SNNs, compared to traditional computing frameworks such ANN.

2.1. SNN and local learning rule

A review of the method and its foundational theories is presented in applying the SNN architecture and learning rule [13]. SNNs, as brain-mimicking computational models that effectively emulate neural circuitry in the human brain, in which the neurons communicate via spikes. Given the prevalent use of LIF neurons in robotics and engineering [5], our focus here is on developing a recurrent EBN comprising these neurons from the perspective of EBN and spike coding theory. To develop an SNN capable of representing temporal variation of parameters such as $\mathbf{x}$, two key assumptions are necessary [13, 14, 16]. First, as it has been expressed in equation (1), using a linear decoder matrix $\mathbf{D}$, it is needed for $\mathbf{x}$ to be extracted from the filtered spike trains.

$$\hat{\mathbf{x}} = \mathbf{D}\mathbf{r}. \tag{1}$$

In this equation, $\mathbf{r} \in \mathrm{R}^N$ represents the filtered spike trains, which are the convolution of spike trains with synaptic kernels, and can be interpreted as instantaneous firing rate [13], and exhibit slower dynamics compared to $\mathbf{s} \in \mathrm{R}^N$. The dynamics of these filtered spike trains obeys the following expression:

$$\dot{\mathbf{r}} = -\lambda\mathbf{r} + \mathbf{s}. \tag{2}$$

The second assumption is that the network minimizes the cumulative error between the actual value of $\mathbf{x}$ and the estimated $\hat{\mathbf{x}}$, focusing on optimizing spike timing rather than changing the output kernel values $\mathbf{D}$. So, the network minimizes the cumulative error between the state and its estimate while limiting computational cost by controlling spike occurrence. To achieve this, it minimizes the following cost function [13, 14]:

$$J = \frac{1}{t}\int_0^t \left(\|\mathbf{x}(\tau) - \hat{\mathbf{x}}(\tau)\|_2^2 + \nu\|\mathbf{r}(\tau)\|_1 + \mu\|\mathbf{r}(\tau)\|_2^2 \right) \mathrm{d}\tau. \tag{3}$$

Here, $\|\cdot\|_2^2$ represents the Euclidean norm, and $\|\cdot\|_1$ indicates $L1$ norms. This firing rule ensures that each neuron emits a spike only when it contributes to reducing the predicted error, a concept akin to predictive coding [14]. The network's neurons have membrane potentials and firing thresholds, described by equations (4) and (5):

$$\boldsymbol{\sigma}_i(\mathrm{t}) = \mathbf{D}_i^{\mathrm{T}}\left(\mathbf{x}(t) - \widehat{\mathbf{x}}(t)\right) - \mu \mathbf{r}_i \tag{4}$$

$$\mathbf{T}_i = \left(\mathbf{D}_i^{\mathrm{T}}\mathbf{D}_i + \nu + \mu\right)/2. \tag{5}$$

Here, $\boldsymbol{\sigma}_i \in \mathrm{R}$ denotes the membrane potential of ith neuron. $\mathbf{D}_i$ is the ith column of the matrix $\mathbf{D}$, reflecting the neuron's output kernel and the change in the error due to a spike of ith neuron. The parameters ν and μ control the trade-off between computational cost and accuracy, with ν encouraging the network to use as few spikes as possible, while the quadratic term is influenced by μ, which is responsible for a uniform distribution of spikes among network neurons. Proper tuning of these parameters yields spiking patterns where neural activity is distributed approximately evenly among neurons. This firing rule operates based on the concept that the network minimizes the cost function by controlling spike occurrence in response to excitation and inhibition, ultimately, reducing the prediction error [13, 14]. After differentiating and simplification of equation (4), the dynamics of membrane potential for a SNN that can mimic any arbitrary dynamics like $\dot{\mathbf{x}} = f(\mathbf{x}) + \mathbf{c}(t)$ has been achieved [13]:

$$\dot{\boldsymbol{\sigma}} = -\lambda\boldsymbol{\sigma} + \mathbf{D}^{\mathrm{T}}\mathbf{c} - \left(\mathbf{D}^{\mathrm{T}}\mathbf{D} + \mu\mathbf{I}\right)\mathbf{s} + \mathbf{D}^{\mathrm{T}}\left(\lambda\mathbf{x} + f(\mathbf{x})\right). \tag{6}$$

The above expression shows that the optimal weight matrix for fast connections is $\boldsymbol{\Omega}_f = \mathbf{D}^{\mathrm{T}}\mathbf{D} + \mu\mathbf{I}$ and also, the optimal encoding matrix is $\mathbf{D}^{\mathrm{T}}$. As it is stated in [13], to be able to implement any arbitrary dynamics in this framework, two approximations are made. Firstly, inspired by theories of adaptive nonlinear control, the last term $\lambda\mathbf{x} + f(\mathbf{x})$ is approximated by a weighted sum of basis functions such as $\lambda\mathbf{x} + f(\mathbf{x}) = \sum_i \boldsymbol{\Omega}_i \phi_i(\mathbf{x})$, where $\boldsymbol{\Omega}_i$ is ith column of $\boldsymbol{\Omega}$ and $\phi_i(.)$ can be any sigmoidal nonlinearity that here is set to $\tanh(\square)$. Second, based on equation (1), the state x is replaced by its estimate $\mathbf{Dr}$, culminating in a refined expression for the dynamics of membrane potential, as follows:

$$\dot{\boldsymbol{\sigma}} = -\lambda\boldsymbol{\sigma} + \mathbf{D}^{\mathrm{T}}\mathbf{c} - \boldsymbol{\Omega}_{\mathrm{f}}\mathbf{s} + \mathbf{D}^{\mathrm{T}}\boldsymbol{\Omega}^{\mathrm{T}}\phi(\mathbf{Dr}). \tag{7}$$

Biologically, in the above expression, the last term corresponds to the nonlinear and highly structured dendrites that perform nonlinear operations in the biological brains. So, to relax the constraints that are caused by the fact that, while the dendrites are often nonlinear their inputs are stochastic, the term $\phi(\mathbf{Dr})$ is replaced by $\psi_i(\mathbf{r}) = \phi_i\left(\mathbf{M}_i^{\mathrm{T}}\mathbf{r} + \theta_i\right)$ which is a highly unstructured nonlinear dendrite that can receive stochastic inputs from other neurons. Where $\mathbf{M}$ is the random fixed matrix, and θ_i is a parameter that determines where the tanh (.) changes its sign in a dendritic branch. This substitution is justified by the fact that the internal dynamics of the network are highly robust to great amounts of noise when the inequality $N > 2\,\mathrm{K}$ is met [13]. The network dynamics are thus defined by the following expression:

$$\dot{\boldsymbol{\sigma}} = -\lambda\boldsymbol{\sigma} + \mathbf{Fc} - \boldsymbol{\Omega}_{\mathrm{f}}\mathbf{s} + \boldsymbol{\Omega}_{\mathrm{s}}\psi(\mathbf{r}). \tag{8}$$

Here, λ represents the leak rate for the membrane potential, and $\mathbf{F} = \mathbf{D}^{\mathrm{T}}$ encodes the transformation of dynamics external stimulation into a spike sequence interpretable by the network. The synaptic weights, $\boldsymbol{\Omega}_{\mathrm{s}}$ and $\boldsymbol{\Omega}_{\mathrm{f}}$ correspond to slow and fast connections, respectively. Slow connections typically govern the implementation of the desired system dynamics, whereas fast connections are crucial for uniformly distributing spikes across the network [13–15]. To incorporate a local learning rule based on adaptive control theories, the error $\mathbf{e}(t) = \mathbf{x}(t) - \widehat{\mathbf{x}}(t)$ is not fed back into the whole network via the input $\mathbf{c}(t)$. Instead, the error is directly injected into each neuron using the optimal encoder $\mathbf{D}^{\mathrm{T}}$. Consequently, with such neuron-specific error feedbacks, the network equation is reformulated as:

$$\dot{\boldsymbol{\sigma}} = -\lambda\boldsymbol{\sigma} + \mathbf{Fc} - \boldsymbol{\Omega}_{\mathrm{f}}\mathbf{s} + \boldsymbol{\Omega}_{\mathrm{s}}\psi(\mathbf{r}) + k\mathbf{D}^{\mathrm{T}}\mathbf{e}. \tag{9}$$

As indicated earlier, $\boldsymbol{\Omega}_{\mathrm{s}}$ plays a crucial role in executing the desired dynamics, necessitating its learning from example trajectories of an external unknown dynamic in a teacher-student fashion. To this aim, the following expression from [13] outlines this learning process:

$$\dot{\boldsymbol{\Omega}}_{\mathrm{s}} = \eta\psi(\mathbf{r})\left(\mathbf{D}^{\mathrm{T}}\mathbf{e}\right)^{\mathrm{T}}. \tag{10}$$

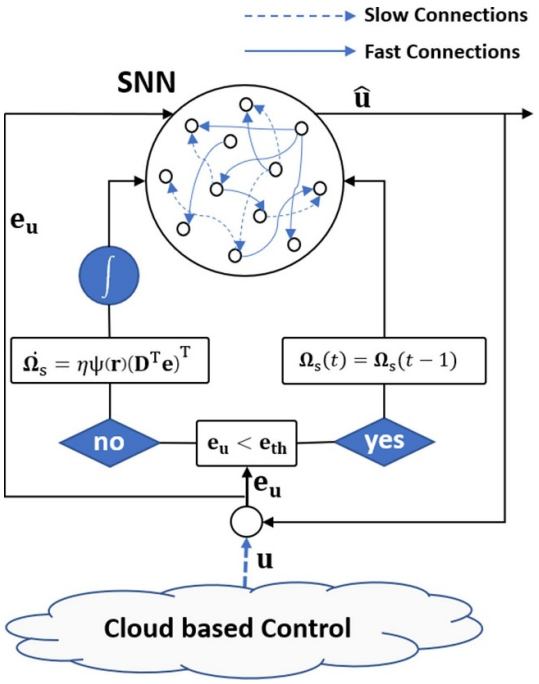

Figure 2. Block diagram of the constrained weight update rule.

This expression demonstrates that the proposed weight update relies on pre-synaptic inputs, which will be provided via dendritic nonlinearities, and the projection of network error both from the last time step of implementation. Therefore, it can be inferred that the learning rule is both spatially and temporally localized [7, 13].

2.2. Online supervised learning architecture

As depicted in figure 2, the architecture for online supervised learning of SNNs utilizes a specific learning rule where the SNN, positioned on the plant, receives a control signal from cloud-based algorithms. The SNN's objective is to learn and replicate this signal until the error $\mathbf{e}_u = \mathbf{u} - \hat{\mathbf{u}}$ drops below a predefined threshold $\mathbf{e}$th. Once this threshold is reached, the SNN continues to control the plant independently, without further weight updates. Here, $\mathbf{u}$ represents the cloud-provided control signal, and $\hat{\mathbf{u}}$ denotes the SNN-reproduced signal. It is notable that, in the considered strategy, the SNN-based edge controller maintains communication with the cloud for the initial 50-time steps. Subsequently, in the post-learning time period, the edge-controller reduces communication with the cloud for all the next time steps, and to avoid the large deviations of $\hat{\mathbf{u}}$ from the desired signal $\mathbf{u}$, every 5-time steps, the SNN-based controller recieves the signal $\mathbf{u}$ from the cloud to update $\mathbf{e}_u$ then checks the threshold crossing for the weight update with the updated error $\mathbf{e}_u$. In this manner, for any practical setup, after 50 initial time steps, the required energy for communication and data transfer between cloud and the edge will be reduced by the amount of almost 80%.

A key aspect of this approach is the SNN's independence from both the source of the control signal and the target plant. This means the SNN requires no training about the adopted controller on the cloud or the dynamical plant, enabling easy implementation on various plants with different control methods.

3. Numerical simulations

In this section, we first apply the proposed framework to a linear workbench problem, evaluating its performance in terms of accuracy and energy efficiency. We then extend the analysis to a real-world scenario, focusing on the optimal control of satellite rendezvous maneuvers.

3.1. Linear workbench system

The performance investigation begins with applying the proposed strategy to the following linear system:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{11}$$

$$\mathbf{u} = -\mathbf{K}_c\mathbf{x} \tag{12}$$

Table 1. Linear system simulation parameters.

Parameter	Value
$\mathbf{x}_0$	$[5,2]^{\mathrm{T}}$
$\mathbf{Q}_c$	$(1e-6)\,\mathrm{I}_6$
$\mathbf{R}_c$	I_3
$\mathbf{K}_c$	$[0.2361,\ 1.2133]^{\mathrm{T}}$
N	30
eth	$(1,1)/10$
λ	0.001
μ	0.001
ν	0.001
k	500
η	0.001

where:

$$\mathbf{A}=\begin{bmatrix}0 & 1\\ -2 & 0\end{bmatrix};\mathbf{B}=\begin{bmatrix}0\\ 1\end{bmatrix}. \tag{13}$$

In this system, $\mathbf{K}_c$ represents the controller gain, designed using LQR optimal control theories. Thus, the $\mathbf{K}_c$ needs to be designed to minimize the cost function, J_c, in equation (14) [21]:

$$J_c=\int_0^{\infty}\left(\mathbf{x}^{\mathrm{T}}\mathbf{Q}_c\mathbf{x}+\mathbf{u}^{\mathrm{T}}\mathbf{R}_c\mathbf{u}\right)\mathrm{d}t. \tag{14}$$

In the above expression, $\mathbf{Q}_c$ and $\mathbf{R}_c$ are the weight matrices that can be determined through trial and error satisfying the conditions $\mathbf{Q}_c > 0$ and $\mathbf{R}_c \geqslant 0$. Finally, the LQR controller gain can be computed using $\mathbf{K}_{\mathrm{LQR}}=\mathbf{R}_c^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{S}$, where $\mathbf{S}$ is the unique positive semidefinite solution of the algebraic Riccati equation:

$$\mathbf{A}^{\mathrm{T}}\mathbf{S}+\mathbf{S}\mathbf{A}-\mathbf{S}\mathbf{B}\mathbf{R}_c^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{S}+\mathbf{Q}_c=0. \tag{15}$$

After, designing the LQR controller, simulations ran for a 10 sec duration with a 0.1 sec timestep, employing parameters from table 1. The decoding matrix $\mathbf{D}$'s elements were sampled from a zero-mean Gaussian distribution with a covariance of 10. It is noteworthy that the tunning parameters of the proposed network structure in this research such as the mentioned Gaussian distribution variance, λ, μ, ν, η, k and **e**th have been tuned using a trial-and-error fashion based on the imposed limitations regarding our design which is a trade-off between accuracy and energy efficiency. Some investigatins on the parameters μ, and ν have been provided in our previous work [15].

Figure 3 presents the controlled states obtained from the physical plant under the SNN-reproduced signal $\widehat{\mathbf{u}}$ for $N=5$, 15 and 30, compared with states from the cloud's dynamical model using the LQR-provided control signal $\mathbf{u}$. Figure 3(a) reveals that the obtained results for the state x_1 of the physical plant using $\widehat{\mathbf{u}}$ have different accuracies for various N. The result for $N=5$ exhibits some deviation from the cloud-provided state, indicating that larger N values enhance tracking accuracy. Increasing the N has increased the tracking accuracy in a way that for $N=30$, the result from the physical plant almost aligns with the dynamic model on the cloud, considering the expected values for x_1. Thus, it can be deduced that the network size influences controller accuracy. Also, the physical plant satisfactorily follows the expected state trajectory using $\widehat{\mathbf{u}}$. Figure 3(b) demonstrates similar observations for the state x_2. The zoomed inset highlights deviations from the desired values, confirming a 5 times higher deviation for $N=5$ in the considered area, compared to $N=30$ for both the states x_1 and x_2. These observations underscore the impact of network size on controller accuracy and the effective tracking of desired state trajectories.

Figure 4 compares the control input $\widehat{\mathbf{u}}$ reproduced by the SNN with the original input $\mathbf{u}$ provided by the cloud, albeit with noticeable fluctuations, which vary depending on number of neurons, N. These fluctuations are more pronounced in regions where the desired signal exhibits higher nonlinearity. Specifically, with a neuron count $N=5$, the fluctuation domain is significantly larger between 0.54 sec compared to the behavior of SNN tracking signals after $t=8$ s, where it significantly dampens. The inset shows the region where the signal changes the rising to declining trends, corresponding to the most nonlinear region of the curve. It becomes clear that an increase in neuron count N leads to improved tracking performance by the SNN, characterized by reduced fluctuation domains in these critical regions. For instance, in the magnified area, the fluctuation domain at $N=5$ is about 10.5 times larger than at

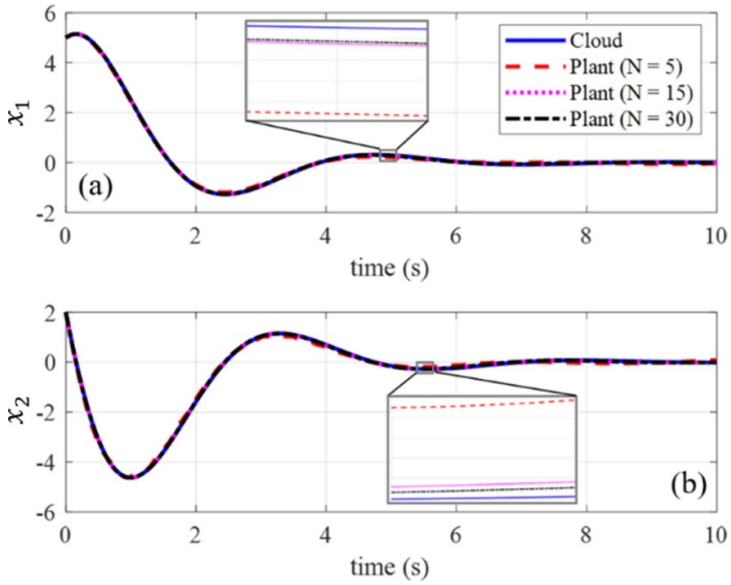

Figure 3. Time history of controlled states obtained from the plant for $N = 5$, 15 and 30 compared with expected states provided by cloud for (a) state x_1 and (b) state x_2.

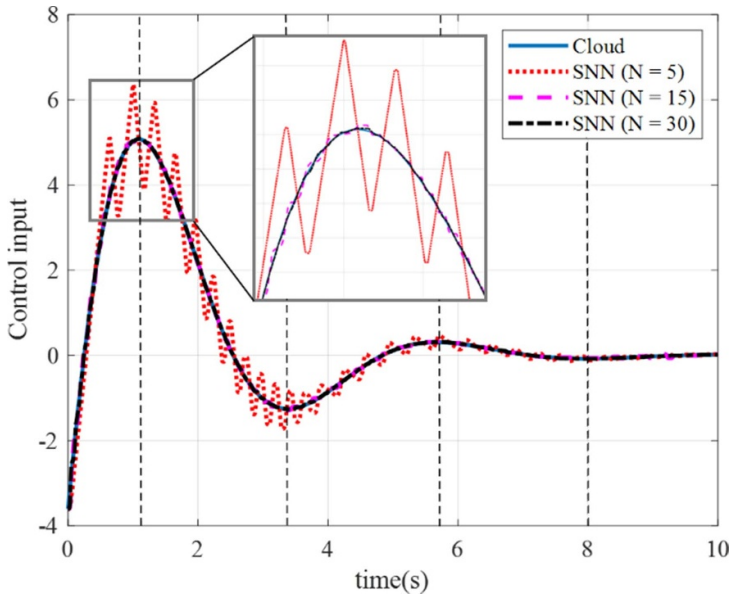

Figure 4. Time history of control input reproduced by SNN on the plant for $N = 5$, 15 and 30 compared with expected input provided by the cloud. Notice, that vertical dashed lines specify where the behavior of the desired signal changes.

$N = 15$, and nearly triple that of $N = 30$. This observation suggests that a higher neuron count is required to respond to any abrupt maneuver, enhancing the accuracy of the SNN's signal reproduction relative to the cloud-provided signal, and consequently improving the control performance of the system. For clarity, figure 4 only displays the results for $N = 5$, 15 and 30.

Further investigation into the effect of neuron count on accuracy is presented in figure 5. In figure 5(b), the variation of normalized tracking error reconfirms that an increase in N leads to enhanced SNN performance, with a notable normalized error reduction of 96%. However, beyond $N = 30$ (Region 2 in the figure), the increase in neuron count does not significantly affect accuracy, which remains nearly constant for the remaining simulation cases. This behavior reflects the intrinsic network scalability of SNNs. This means that they can maintain a redundant resource of spiking neurons that ideally do not consume energy when not in use, remaining in standby mode and ready to engage in data processing for more complex conditions as needed. Thus, for the conventional design, it is better to consider more neurons than the N which

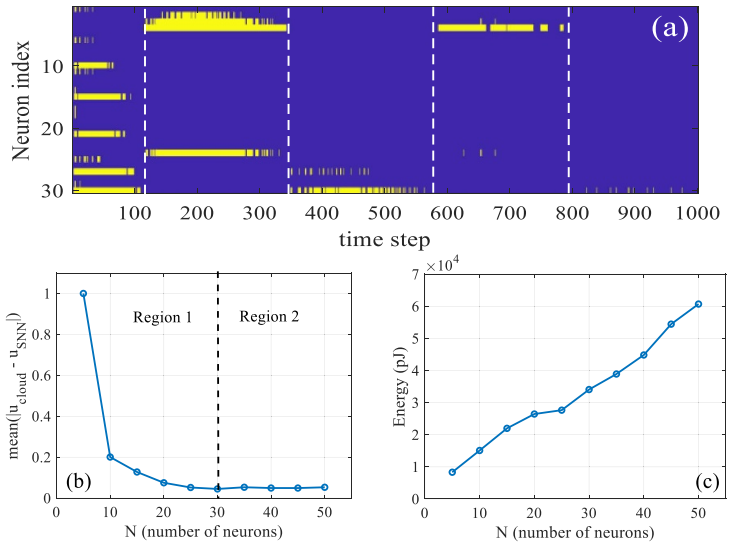

Figure 5. (a) Spiking activity as a function of time for 30 neurons, vertical dashed lines specify where the spiking pattern regime changes (b) normalized tracking error versus N, (c) energy consumption vs. N.

corresponds to minimal error to maintain neuron silencing in the neuromorphic system for the conditions in which the SNN encounters more complex problems.

Figure 5(a) demonstrates that the SNN, in accordance with the EBN theory, operates with minimal spike usage. In this problem, the SNN utilized 1444 spikes during the simulation period, which is only 4.81% of the potential 30 000 spikes (for 30 neurons across 1,000 time steps). With an energy consumption of 23.6 pJ per spike in neuromorphic hardware (e.g. intel's Loihi neuromorphic chip) [27], the SNN would consume only 3.4078 $\times$ 104 pJ of energy for the task considered here and this is much more efficient in terms of energy consumption.

Moreover, it is noticeable that each vertical dashed line on the spiking pattern corresponds to the vertical dashed lines exhibited in figure 4. It is clearly apparent that when the graph of desired input presented in figure 4 passes from the vertical dashed lines, the changes in the direction of desired values cause the change in the spiking activity regime between the vertical dashed lines demonstrated in figure 5(a). Thus, it can be deduced that corresponding to any change in the desired value, the SNN adjusts its neural activity.

Figures 5(b) and (c) demonstrate the changes in the tracking accuracy of the SNN and its energy consumption versus the number of neurons respectively. The obtained results, reconfirm that the reduction of number of neurons will lead to declination in energy consumption of the SNN. They demonstrate that for the considered system in this part, we can reduce the number of neurons from 50 to 30 to decline the energy consumption without experiencing significant change in the tracking accuracy in region 2 of figure 5(b), but the further reduction of number of neurons below 30 in Region 1 of figure 5(b) will lead to energy efficiency at the cost of network tracking accuracy. Moreover, from the perspective of neuron silencing which is sometimes an inevtiable condition in practice, figure 5(b) and is demonstrating that in the condition of neuron silencing in which the neuromorphic chip will lose some of its neurons, the proposed architecture will keep working without any signifucant loss in its accuracy from $N = 50$ to $N = 20$.

Figure 6 illustrates the weight update of a single neuron (elements of the 1st row of $\mathbf{\Omega}_s$) for two different cases of $N = 10$ and $N = 50$ over the simulation time period. In the presented graphs in figure 6 each line corresponds to the temporal variations of a synaptic weight. Figures 6(a) and (b) reveal the temporal variations for the cases of $N = 10$ and $N = 50$ respectively. The depicted graphs confirm that, owing to the update rule presented in figure 2, for both cases the weight update rule was not implemented continuously over the simulation time. In contrast to what it has been shown for the case of $N = 50$ which its weights have immediately converged to a constant set and it continues without any weight update until the end of the simulation, for the case of $N = 10$ the update rule is implemented continuously for the first 50 time steps (until $t = 0.5$s), then it continued with constant weights until almost $t = 1.8$s which it has updated its weights that demonstrate the proposed framework encountered to a tracking error threshold crossing. Furthermore, a comparison between these figures demonstrates that, in comparison with the obtained results for the case of $N = 10$, although the synaptic weights have converged to constant values in almost $t = 0.5$s, for the case of $N = 50$ the synaptic weights exhibited the faster convergence to constant values I which they have converged in a time less than $t = 0.1$s. therefore, it can be inferred that, increasing the

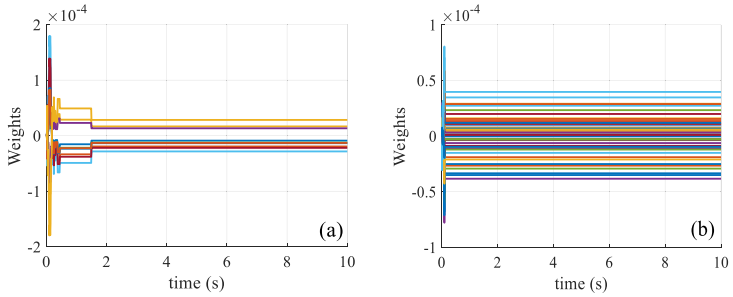

Figure 6. Temporal variation of synaptic weights of a single neuron (elements of the 1st row of $\mathbf{\Omega}_s$) for the case of (a) $N = 10$ and (b) $N = 50$.

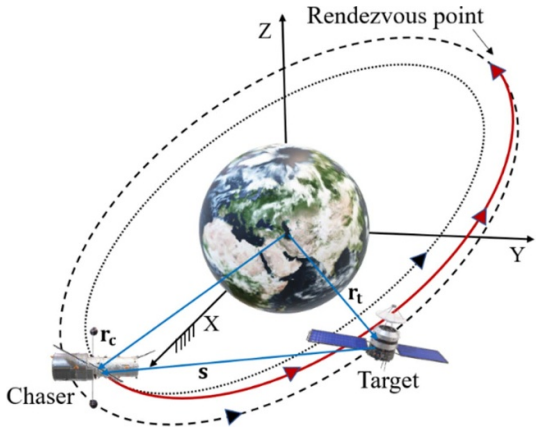

Figure 7. Schematic of satellite rendezvous problem. By permission from [15], Reproduced from [28]. CC BY 4.0.

number of neurons will increase the rate of convergence of synaptic weights. Besides, figure 6 shows the ability of the proposed controller on the edge to work without continous communication and weight update per each time step. Moreover, the graphs depict almost symmetrical patterns around zero, which indicates that the level of excitation (E: positive weights) is approximately equal to the level of inhibition (I: negative weights) in the EBN which is an evidence of having one of the necessary conditions for E–I balance [13]. It is notable that, we are not using the standard definition of E–I balance, which is based on the balance of excitatory and inhibitory synaptic inputs to a neuron; instead, we rely indirectly on the symmetry of the sign of the learned weights to hint at E–I balance.

Conclusively, the results in this section have demonstrated the ebergy efficiency of the proposed framework. Notably, it excels in accuracy and significantly mitigates the need for continuous plant-cloud communication during all operational steps. This reduction in data transfer between the plant and cloud substantially lowers energy consumption, a crucial benefit in today's energy-constrained systems. Additionally, the findings underscore that, unlike conventional methods, the integration of neuromorphic architectures, owing to the scalability of the network and energy efficiency of SNNs, facilitates more reliable and energy-efficient control systems on the plant. Moreover, the versatility of our proposed framework allows for its easy adaptation to centralized control scenarios in dynamical systems manipulation.

3.2. Satellite rendezvous without obstacle

In this section, the proposed framework has been applied to the problem of satellite rendezvous as one of the fundamental challenges for various space missions, including assembling, servicing, or refueling spacecraft in orbit [22]. As it has been shown in figure 7, the satellite rendezvous is a multi-object maneuver between two satellites in which the chaser satellite approaches the target and reaches together on the rendezvous point.

This study aims to implement the LQR controller on the cloud, utilizing CW equations. First, we will introduce the state space model for the satellite rendezvous problem to facilitate the development of the LQR controller. Following this, we will present the simulation results and discussions to demonstrate the effectiveness of the implemented controller.

Table 2. Paremeters for satellite rendezvous.

Parameter	Value
$\mathbf{r}_0$ (m)	$[70, 30, -5]^{\mathrm{T}}$
$\mathbf{v}_0$ $\left(\mathrm{m\,s^{-1}}\right)$	$[-1.7, -0.9, 0.25]^{\mathrm{T}}$
$\mathbf{x}_0$	$[\mathbf{r}_0, \mathbf{v}_0]^{\mathrm{T}}$
μ_{earth} $\left(\mathrm{km^3\,s^{-2}}\right)$	398600
R_o (km)	6771
N	50
$\mathbf{e}_{\mathrm{th}}$	$(1, 1, 1)/10000$
λ	0.0001
μ	0.0001
ν	0.0001
k	250
η	0.001

The following system of differential equations represents the linearized CW equations which describe the relative motion in the target frame [24]:

$$\ddot{x} - 2n\dot{z} = f_x \tag{16}$$

$$\ddot{y} + n^2\dot{y} = f_y \tag{17}$$

$$\ddot{z} + 2n\dot{x} - 2n^2\dot{z} = f_z. \tag{18}$$

Here, $n = \sqrt{\mu_{\mathrm{earth}}/R_{\mathrm{o}}^3}$, where μ_{earth} is the earth's gravitational parameter, and R_{o} represents the orbital radius of the target spacecraft, and n is the mean motion. Considering the state vector as $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^{\mathrm{T}}$, an input vector $\mathbf{u} = [u_1, u_2, u_3] = [f_x, f_y, f_z]$, the state space form of the CW equations is:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{19}$$

where matrices **A** and **B** are given by the following equations:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2n \\ 0 & 0 & 0 & 0 & -n^2 & 0 \\ 0 & 0 & 0 & -2n & 0 & 2n^2 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{20}$$

Generally, for any pair $(\mathbf{A}, \mathbf{B})$ which met the controllability condition, the LQR control input can be computed using the predecure presented in previous section. After designing the controller gain, we integrated the cloud-based control system for satellite rendezvous as shown in figure 1. This integration utilized the strategy outlined in figure 3, employing an SNN with N neurons in the proposed cloud-based online supervised learning architecture. In this scenario, the chaser satellite acts as the desired plant, receiving the control signals generated by the SNN.

To validate our approach, we conducted simulations, based on parameters from table 2, over a time duration of 360 s with a time step of 0.1 s.

It is important to note that in these simulations, the elements of decoding matrix **D** are sampled from a zero-mean Gaussian distribution with a covariance of 1/1000. Figure 8 reveals the trajectories for the case of satellite rendezvous without obstacles. The comparison between the obtained trajectory from the plant with the trajectory expected from the cloud in the ideal condition confirms that the trained SNN could satisfactorily perform the considered task. Figure 9 demonstrates the proposed framework, and how effectively it controls the states of the chaser satellite. The satellite's actual performance, guided by the SNN-generated control inputs, closely followed the expected state values calculated on the cloud.

To provide a quantitative comparison for the effectiveness of the proposed method and obtained accuracy from the SNN-based controller and the LQR on the cloud in the ideal condition, table 3 demonstrates the obtained mean on the erros for the last 60 s of the simulation after $t = 300$s. although the obtained errors for the SNN-based framework are greater by small amounts but they are in the same order

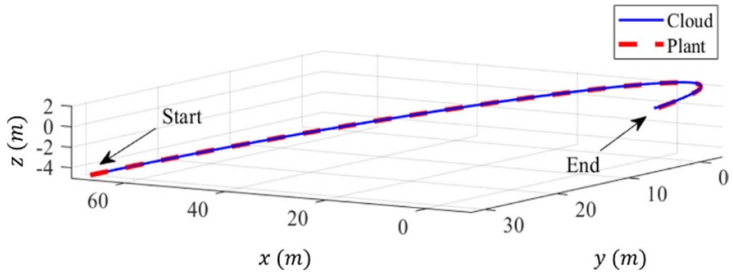

Figure 8. Comparison of obtained trajectory in 3D space for the plant under the SNN control with the expected trajectory provided by cloud for satellite rendezvous without obstacle. Note: the supplementary video provides a visual demonstration of the satellite's control with respect to spiking activities and energy consumption in figure 11.

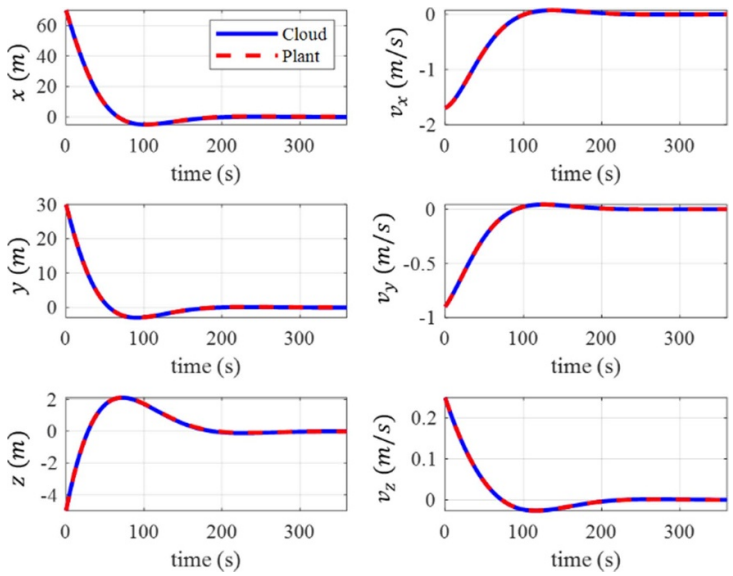

Figure 9. Time histories of obtained results for the controlled states from the plant (red dashed line) in comparison with cloud (blue solid line) for satellite rendezvous without obstacle.

Table 3. Obtained errors for the different controllers in the presence of dynamic obstacle.

State	LQR	SNN
$x\,(\mathrm{m})$	0.0224	0.0568
$y\,(\mathrm{m})$	0.0058	0.0508
$z\,(\mathrm{m})$	0.0049	0.0945
$v_x\,(\mathrm{m\,s^{-1}})$	0.0012	0.0018
$v_y\,(\mathrm{m\,s^{-1}})$	5.5356×10^{-04}	4.5175×10^{-04}
$v_z\,(\mathrm{m\,s^{-1}})$	5.1916×10^{-04}	4.1582×10^{-04}

with the obtained results for the LQR in the ideal condition that is an acceptable performance for the SNN-based framewrok in the terms of effectiveness and accuracy.

Figures 10(a)–(c) illustrate the results for the control input $\hat{\mathbf{u}}$ elements reproduced by the SNN on the plant, compared with input $\mathbf{u}$ provided by the cloud. These figures show that the SNN closely tracked the cloud-provided signal across all input vector elements, suggesting effective control of the physical plant. Figures 10(d)–(f) show the tracking errors corresponding to each input element for the first 5 sec. This early time frame is highlighted for a clear visual representation of error convergence. Notably, the errors converge to nearly zero around 2 sec and the system maintains minimal fluctuations thereafter. This rapid error convergence suggests the SNN's high responsiveness and precision in mimicking the control inputs, underlining its suitability for dynamic and time-sensitive applications like satellite rendezvous.

Figure 11(a) shows the spiking pattern of the SNN for the given task, demonstrating efficient task execution with minimal spike usage. The spiking pattern indicates three distinct spiking regimes corresponding to changes in control input behaviors: a shift in input direction around $t = 40$s and input

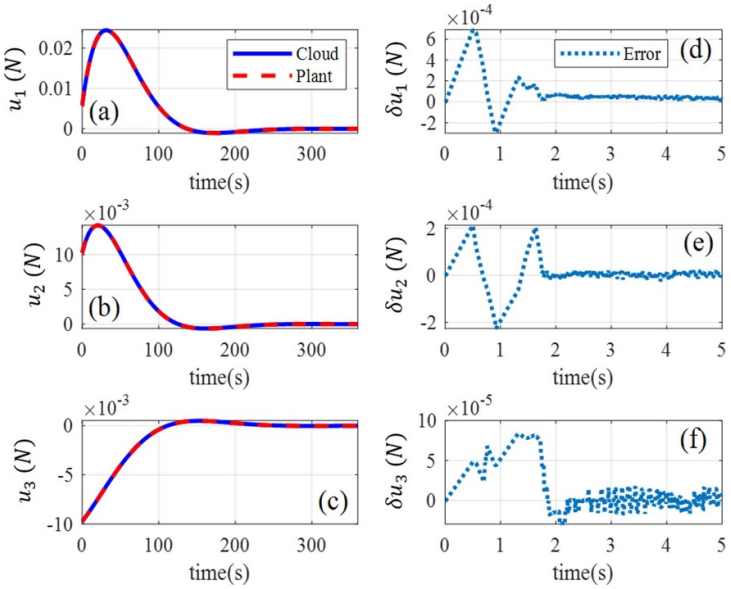

Figure 10. Comparative time histories and error analysis in satellite rendezvous without obstacle. This figure presents a detailed comparison of the control signals as reproduced by the satellite system against those provided by the cloud. Subplots (a) to (c) depict the control input vectors, highlighting the alignment and accuracy of the satellite system in following the cloud directives in an obstacle-free environment. Subplots (d) to (f) focus on error convergence, showcasing the system's precision and ability to closely match the cloud's control signals, illustrating its energy efficiency in a clear trajectory scenario.

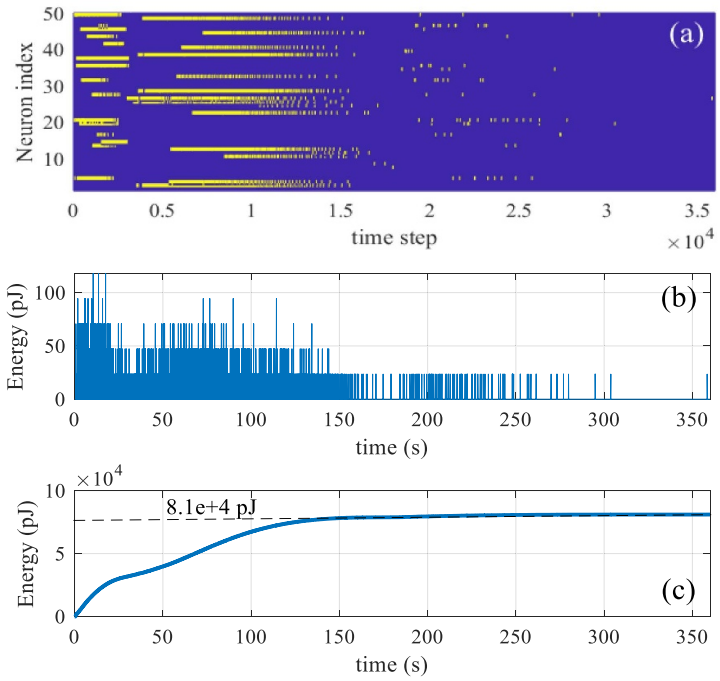

Figure 11. (a) Spiking activities in SNN, (b) energy consumed per time step, and (c) accumulative energy consumption for satellite rendezvous without obstacle.

convergence to near zero at $t = 200$s. The SNN emitted 3434 spikes across 50 LIF neurons, just 0.19% of the potential 1800 000 spikes. Figure 10(b) shows the temporal variation of the consumed energy per time step for spike emission. It reveals that most of the energy is consumed before $t = 150$s where the elements of the input vector are significantly greater than zero and after $t = 150$s, where the input vector elements are almost near to zero the consumed energy is considerably reduced, and this is vividly apparent in the obtained spiking pattern.

Figure 11(c) demonstrates the energy consumption versus time for performing the considered task. It exhibits that the energy consumption has drastically increased to 8.1×10^4 pJ at $t = 150$s, then it remained

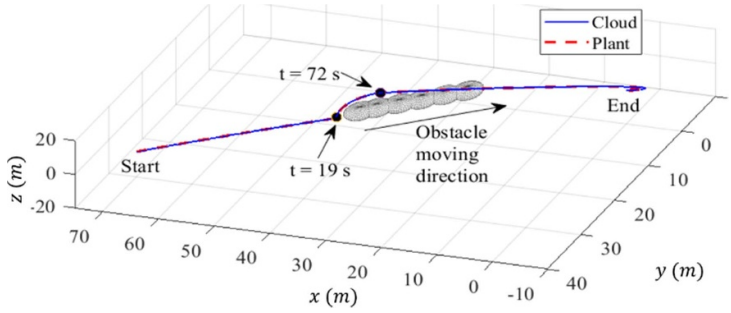

Figure 12. Comparison of obtained trajectory in 3D space for the plant under the SNN control with the expected trajectory provided by cloud for satellite rendezvous with a dynamic obstacle. Note: the supplementary video provides a visual demonstration of the satellite's navigation for dynamic obstacle avoidance with respect to spiking activities and energy consumption in figure 15.

almost constant. This pattern of energy usage underscores the SNN's energy efficiency, particularly in the latter stages when control inputs stabilize and all LIF neurons are in standby mode. Note that the energy consumption analysis is based on the characteristics of Intel's Loihi chip, a benchmark for energy-efficient neuromorphic computing.

3.3. Satellite rendezvous with dynamic obstacle

This section presents the results of our framework's performance in scenarios involving dynamic obstacles, building upon the previous section which focused on static obstacles. Figure 12 showcases the trajectory of a satellite rendezvous maneuver in the presence of a dynamic obstacle within a three-dimensional space. The figure indicates that the trajectory followed by the satellite, as directed by the SNN, aligns closely with the expected trajectory computed by the cloud. Our findings demonstrate that the satellite successfully executed the obstacle avoidance maneuver. This maneuver began at $t = 19$s and continued until $t = 27$s, effectively navigating around the dynamic obstacle. Notably, a supplementary video file is available, providing a clear visual representation of the satellite's navigation during the obstacle avoidance maneuver. This video vividly illustrates the satellite's response to the dynamic obstacle, offering a more comprehensive understanding of the framework's capabilities in real-time scenarios. This detailed view offers valuable insights into the effectiveness of the SNN in ensuring both the successful execution of the planned path and the avoidance of dynamic obstacles, thereby illustrating the practical applicability and robustness of our framework in dynamic space navigation scenarios.

Figure 13 further elaborates on the dynamics of the satellite's interaction with the moving obstacle. It graphically represents the control inputs and trajectory adjustments as the satellite navigates the dynamic obstacle in real time. This figure provides a detailed analysis of how the satellite, under the guidance of the SNN-based controller, dynamically adjusts its path to maintain the planned trajectory while avoiding obstacles. Similar to the previous case of a static obstacle, the obtained results demonstrate the SNN's adaptability and precision in managing complex, changing scenarios, affirming its capability to modify the satellite's path. The figure highlights key moments of trajectory adjustment with regard to figure 12 and showcases how the control inputs vary in response to the obstacle's movement.

In order to have a quantitative comparison between the obtained accuracies for peoposed architecture and the LQR framework which is almost a standard and widely adopted method in the space robotics, table 4 compares the obtained mean on the control residuals fo the time periof after $t = 300$ in that almost both the frameworks are converged to their final results. Table 4 demonstrates that, although the obtained errors for the SNN-based framework are greater by small amounts but they are in the same order with the obtained results for the LQR in the ideal condition that is an acceptable performance for the SNN-based framewrok in the terms of effectiveness and accuracy.

Figures 14(a)–(c) display the comparison between the control inputs reproduced by the SNN on the satellite and those provided by the cloud. These results exhibit a performance akin to the static obstacle scenario, with the SNN's signals closely tracking those from the cloud. A notable difference in this dynamic obstacle scenario is the extended duration of interaction between the satellite and the moving obstacle, starting at $t = 19$s and continuing until $t = 72$s. This prolongs the obstacle avoidance maneuver by approximately 13.5 s compared to the static obstacle case. Figures 14(d)–(f) present the tracking errors for the control inputs derived from the SNN relative to the cloud's signals. The results show that tracking convergence occurred in less than 2 s, indicating satisfactory performance.

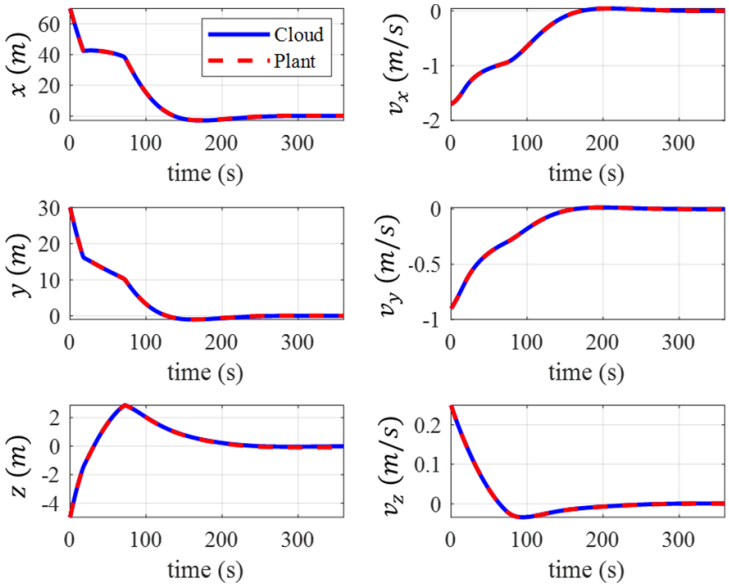

Figure 13. Time histories of obtained results for the controlled states from the plant in comparison with cloud results for satellite rendezvous with dynamic obstacle.

Table 4. Obtained errors for the different controllers in the presence of dynamic obstacle.

State	LQR	SNN
$x\,(\mathrm{m})$	0.0990	0.2720
$y\,(\mathrm{m})$	0.0316	0.0849
$z\,(\mathrm{m})$	0.0301	0.0951
$v_x\,(\mathrm{m\,s^{-1}})$	8.0106×10^{-04}	0.0017
$v_y\,(\mathrm{m\,s^{-1}})$	4.6160×10^{-04}	7.9314×10^{-04}
$v_z\,(\mathrm{m\,s^{-1}})$	5.1709×10^{-04}	1.9276×10^{-04}

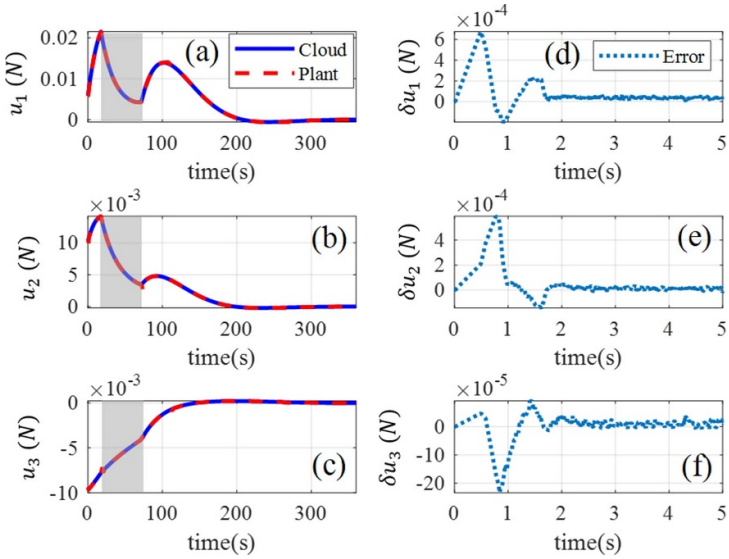

Figure 14. Comparative analysis and error convergence in satellite rendezvous with a dynamic obstacle. This graph presents the time histories of the reproduced control signals versus those provided by the cloud. Subplots (a) to (c) depict the comparison of control input vectors, illustrating the responsiveness of the SNN-based system to the cloud-generated directives. Subplots (d) to (f) focus on error convergence, showcasing the precision and accuracy of the SNN in tracking the provided control signals, and highlighting the efficacy of the system in real-time adjustment and navigation in the presence of a dynamic obstacle.

Figure 15(a) illustrates the spiking pattern of the SNN in the dynamic obstacle scenario. The spiking pattern during the interaction phase is highlighted to facilitate analysis. Compared to the without obstacle scenario in figure 11(a), there's an increase in interaction length and a shift in spiking activity, demonstrating

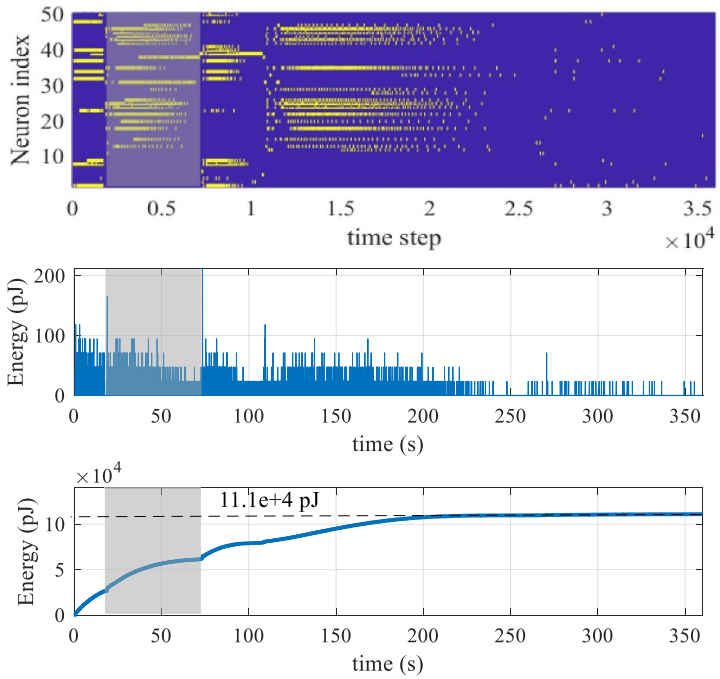

Figure 15. (a) Spiking pattern, (b) energy consumed per time step, and (c) total energy consumption for satellite rendezvous with a dynamic obstacle.

Table 5. Obtained errors for the different controllers in the presence of dynamic obstacle with $\mathbf{\Omega}_s = 0$.

State	SNN	SNN ($\mathbf{\Omega}_s = 0$)
$x\,(\mathrm{m})$	0.2720	1.9420
$y\,(\mathrm{m})$	0.0849	0.7700
$z\,(\mathrm{m})$	0.0951	0.2136
$v_x\,(\mathrm{m\,s^{-1}})$	0.0017	0.0024
$v_y\,(\mathrm{m\,s^{-1}})$	7.9314×10^{-04}	6.3615×10^{-04}
$v_z\,(\mathrm{m\,s^{-1}})$	1.9276×10^{-04}	6.1752×10^{-04}

the SNN's adaptability to changing conditions. In this scenario, the SNN utilized 4704 spikes, about 0.26% of its potential spiking capacity, resulting in an additional energy consumption of 29 972 pJ on Intel's Loihi chip for 1270 additional spikes compared to the scenario without obstacle. Figure 15(b) shows the energy consumption per time step of the SNN in the dynamic obstacle scenario, clearly visible, there is a significant rise in energy use to 170 pJ at $t = 19$s, when the satellite enters the interaction zone, followed by a decrease until the end of the interaction phase. After exiting the interaction zone, the energy consumption spikes again as the SNN adjusts to the new conditions.

Figure 15(c) depicts the cumulative energy consumption, reaffirming the observations from previous analyses. The graph's slope increases at the start and end of the interaction period, indicative of heightened energy use, followed by a steady decline. Around $t = 230$s, the slope stabilizes, reflecting a consistent spiking pattern and convergence of energy consumption at approximately 11.1×10^4 pJ.

Furthermore, at the final step of the presented study, to show the effectiveness of the $\mathbf{\Omega}_s$ we have compared the obtained errors from the simulations of this part (Reported in table 4) with same scenario in the presence of moving obstacle considering $\mathbf{\Omega}_s = 0$. Obtained results are tabulated in table 5. Results demonstrate one-order rise in the errors obtained for the states describing the position of the satellite which is so critical in the space robotics and almost an acceptable performance in the obtained results for the satellite velocity vector which will lead to a small rise in the velocity. Thus, the presence of the $\mathbf{\Omega}_s$ in our proposed network structure is vital.

Moreover, to assess the robustness of the proposed architecture in encountering with real-time neuron silencing which is inevitable in real-world applications, considering this situation, the rendezvous problem in the presence of dynamic obstacle has been repeated. Figure 16 shows the spiking pattern for this scenario. Shadowed areas, demonstrate two stages of neuron silencing. In the region 1, the SNN loses 20 neurons, and in the region 2, the SNN loses 15 of the remaining neurons and the control process continuous with only 15

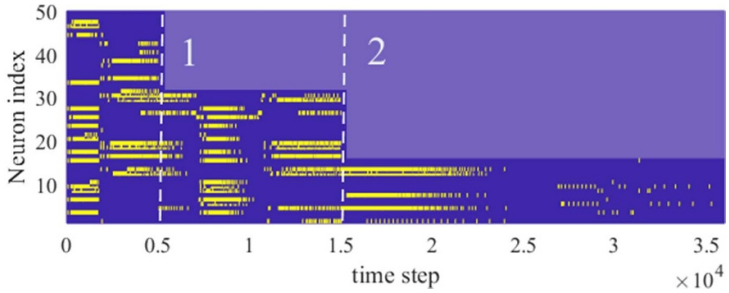

Figure 16. Spiking pattern of obtained for the rendezvous problem in the presence of dynamic obstacle with neuron silencing.

Table 6. Obtained errors for the case studies with-/without- neuron silencing.

State	Without Silencing	With Silencing
$x\,(\mathrm{m})$	0.2720	0.4885
$y\,(\mathrm{m})$	0.0849	0.2359
$z\,(\mathrm{m})$	0.0951	0.1075
$v_x\,(\mathrm{m\,s^{-1}})$	0.0017	0.0017
$v_y\,(\mathrm{m\,s^{-1}})$	7.9314×10^{-04}	4.0112×10^{-04}
$v_z\,(\mathrm{m\,s^{-1}})$	1.9276×10^{-04}	4.3436×10^{-04}

neurons after $t = 150$s. As it has been depicted in figure 16, it is vividly visible that after each time of neuron silencing in two regions, to compensate the lack of neurons, the SNN activates the standby neurons and raised the spiking rates of the active neurons which confirms the robustness of the SNNs in encountering with real-time neuron silencing. Note that, to avoid multiple similar figures, for this case, just the spiking pattern has been provided.

Similar to the previous cases, for a quantitative comparison of the obtained accuracies in the case studies with-/without- real-time neuron silencing table 6 compares obtained accuracies for the time-period after $t = 300$. Compared to the obtained results for the case study without neuron silencing (reported in table 4), results, demonstrate an acceptable accuracy of the proposed structure considering real-time neuron silencing which confirms the robustness of the proposed method to the loss of neurons.

Finally, to assess the proposed discontinuous communication strategy in terms of energy efficiency, the last scenario (real-time neuron silencing in the presence of dynamic obstacle is considered) has been repeated considering continuous and discontinuous communication approach. Figure 17 compares the communication methods. In this figure, '1' refers to the active antenna for communication (data transmission mode), and '0' refers to the time steps in which the antenna is in the standby mode. So, if the edge device uses any conventional antenna with a predefined energy consumption per each communication cycle, our proposed approach will lead to 80% reduction in the required energy for cloud-edge communication.

Furthermore, to check the performance of the proposed method in terms of control accuracy, table 7 compares the accuracy obtained for two different methods of communication. Obtained results demonstrate an acceptable accuracy for the proposed method compared to the continuous communication. Also, the results shows that even real-time neuron silencing cannot affect the performance of our proposed communication strategy.

It is notable that all the reported results in the presented research are obtained from numerical simulations in the MATLAB environment.

4. Conclusion

Addressing the challenge of resource-intensive control systems, our research presents an innovative approach to dynamic system control using SNNs for energy efficiency. Leveraging cloud-edge-based learning, SNNs accurately replicate control signals for autonomous plant operation, reducing the need for constant plant-cloud communication. The architecture demonstrates a 96% reduction in tracking error with enhanced neuron count and adapts computational resources effectively to varying complexities. Notably, energy consumption increases only moderately under challenging conditions. This study underscores the potential of SNNs in real-time applications, offering a pathway for efficient and scalable control systems in robotics and beyond.

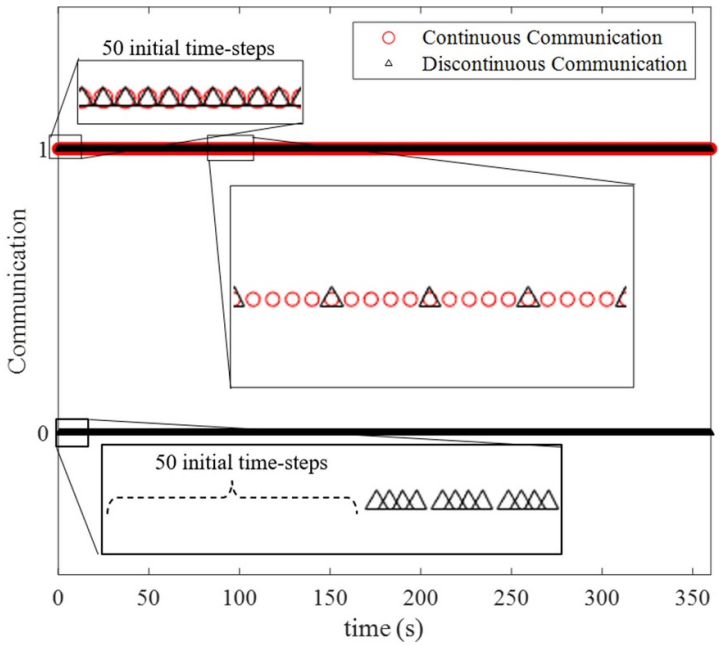

Figure 17. Comparison of proposed discontinuous communication with the.

Table 7. Obtained errors for the case studies with continuous and discontinuous communication.

State	Continuous	Discontinous
$x\,(\mathrm{m})$	0.5631	1.2818
$y\,(\mathrm{m})$	0.2180	0.3572
$z\,(\mathrm{m})$	0.0778	0.0538
$v_x\,(\mathrm{m\,s^{-1}})$	0.0068	0.0015
$v_y\,(\mathrm{m\,s^{-1}})$	0.0028	5.2669×10^{-04}
$v\,(\mathrm{m\,s^{-1}})$	1.5703×10^{-05}	4.1079×10^{-04}

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Conflict of interest

The authors declare that they have no conflicts of interest related to this research. The study was conducted in an objective and unbiased manner, and the results presented herein are based on rigorous analysis and interpretation. The authors have no financial or personal relationships with individuals or organizations that could potentially bias the findings or influence the conclusions of this study.

ORCID iD

Yaser Mike Banad https://orcid.org/0000-0001-7339-810X

References

[1] Schuman C D, Kulkarni S R, Parsa M, Mitchell J P, Date P and Kay B 2022 Opportunities for neuromorphic computing algorithms and applications *Nat. Comput. Sci.* **2** 10–19
[2] Mahmoud M S 2019 Cloud-based control systems: basics and beyond *J. Phys.: Conf. Ser.* **1334** 012006
[3] Schlechtendahl J, Kretschmer F, Sang Z, Lechler A and Xu X 2017 Extended study of network capability for cloud based control systems *Robot. Comput. Integr. Manuf.* **43** 89–95
[4] Maass W 1997 Networks of spiking neurons: the third generation of neural network models *Neural Netw.* **10** 1659–71
[5] Tang G 2022 Biologically inspired spiking neural networks for energy-efficient robot learning and control *Doctoral dissertation* The State University of New Jersey, School of Graduate Studies, Rutgers
[6] Echraghian J K, Ward M, Neftci E O, Wang X, Lenz G, Dwivedi G, Bennamoun M, Jeong D S and Lu W D 2023 Training spiking neural networks using lessons from deep learning *Proc. IEEE* **111** 1016–54
[7] Yamazaki K, Vo-Ho V K, Bulsara D and Le N 2022 Spiking neural networks and their applications: a review *Brain Sci.* **12** 863

[8] Legenstein R, Pecevski D and Maass W 2008 A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback *PLoS Comput. Biol.* **4** e1000180
[9] Wang X, Lin X and Dang X 2020 Supervised learning in spiking neural networks: a review of algorithms and evaluations *Neural Netw.* **125** 258–80
[10] Ponulak F and Kasinski A 2010 Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting *Neural Comput.* **22** 467–510
[11] DeWolf T, Stewart T C, Slotine J J and Eliasmith C 2016 A spiking neural model of adaptive arm control *Proc. R. Soc.* B **283** 2016–34
[12] Bougains A and Shanahan M 2010 Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity *2010 Int. Joint Conf. on Neural Networks (IJCNN)* pp 1–8
[13] Alemi A, Machens C, Deneve S and Slotine J J 2018 Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules *Proc. AAAI Conf. on Artificial Intelligence* vol 32
[14] Boerlin M, Mchanes C K and Deneve S 2013 Predictive coding of dynamical variables in balanced spiking networks *PLoS Comput. Biol.* **9** e10032–58
[15] Ahmadvand R, Sharif S S and Banad Y M 2023 Neuromorphic robust framework for concurrent estimation and control in dynamical systems using spiking neural networks (arXiv:2310.03873)
[16] Slijkhuis F S, Keemink S W and Lanillos P 2022 Closed-form control with spike coding networks (arXiv:2212.12887)
[17] Ahmadvand R, Sharif S S and Banad Y M 2023 Enhancing energy efficiency and reliability in autonomous systems estimation using neuromorphic approach (arXiv:2307.07963)
[18] Kiani M and Ahmadvand R 2022 The strong tracking innovation filter *IEEE Trans. Aerosp. Electron. Syst.* **58** 3261–70
[19] Guez A and Selinsky J 1989 Neurocontroller design via supervised and unsupervised learning *J. Intell. Robot. Syst.* **2** 307–35
[20] Gilra A and Gerstner W 2017 Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network *elife* **6** e28295
[21] Okyere E, Bousbaine A, Poyi G T, Joseph A K and Andrade J M 2019 LQR controller design for quad-rotor helicopters *J. Eng.* **2019** 4003–7
[22] Flores-Abad A, Ma O, Pham K and Ulrich S 2014 A review of space robotics technologies for on-orbit servicing *Prog. Aerosp. Sci.* **68** 1–26
[23] Branz F, Savioli L, Francesconi A, Sansone F, Krahn J and Menon C 2013 Soft docking system for capture of irregularly shaped, uncontrolled space objects *6th European Conf. on Space Debris, ESA/ESOC* (*Darmstadt, Germany*)
[24] Arantes G and Martins-Filho L S 2014 Guidance and control of position and attitude for rendezvous and dock/berthing with a noncooperative/target spacecraft *Math. Problems Eng.* **2014** 1–8
[25] Lemaire E, Cordone L, Castagnetti A, Novac P E, Courtois J and Miramond B 2022 An analytical estimation of spiking neural networks energy efficiency *Int. Conf. on Neural Information Processing* pp 574–87
[26] Ahmadvand R, Sharif S S and Banad Y M 2024 Advancing precision in multi-agent systems: a neuromorphic approach with spiking neural network-modified sliding innovation filter. *Proc. SPIE* **12949** 190–4
[27] Davies M *et al* 2018 Loihi: a neuromorphic manycore processor with on-chip learning *IEEE Micro* **38** 82–99
[28] Ahmadvand R, Sharif S S and Banad Y M 2024 A cloud-edge framework for energy-efficient event-driven control: an integration of online supervised learning, spiking neural networks and local plasticity rules (arXiv:2405.02316)