# Analog Sequential Hippocampal Memory Model for Trajectory Learning and Recalling: A Robustness Analysis Overview

ADVANCED INTELLIGENT SYSTEMS Open Access
www.advintellsyst.com

# Analog Sequential Hippocampal Memory Model for Trajectory Learning and Recalling: A Robustness Analysis Overview

*Daniel Casanueva-Morato,* Alvaro Ayuso-Martinez, Giacomo Indiveri, Juan P. Dominguez-Morales, and Gabriel Jimenez-Moreno*

**The rapid expansion of information systems in all areas of society demands more powerful, efficient, and low-energy consumption computing systems. Neuromorphic engineering has emerged as a solution that attempts to mimic the brain to incorporate its capabilities to solve complex problems in a computationally and energy-efficient way in real time. Within neuromorphic computing, building systems to efficiently store the information is still a challenge. Among all the brain regions, the hippocampus stands out as a short-term memory capable of learning and recalling large amounts of information quickly and efficiently. Herein, a spike-based bio-inspired hippocampus sequential memory model is proposed that makes use of the benefits of analog computing and spiking neural networks (SNNs): noise robustness, improved real-time operation, and energy efficiency. This model is applied to robotic navigation to learn and recall trajectories that lead to a goal position within a known grid environment. The model is implemented on the special-purpose SNNs mixed-signal DYNAP-SE hardware platform. Through extensive experimentation together with an extensive analysis of the model's behavior in the presence of external noise sources, its correct functioning is demonstrated, proving the robustness and consistency of the proposed neuromorphic sequential memory system.**

## 1. Introduction

The expansion of information systems to all areas of society has generated the need for new computing systems that are more powerful, more efficient, and consume less power than current systems.[1–3] Different fields of engineering are exploring new alternatives to solve these problems. Among them, neuromorphic engineering stands out. This field focuses on proposing systems that are inspired by the functioning of the brain, imitating specific parts of it, both at the architecture level or at the functionality level. The brain is capable of solving complex problems efficiently in real time and with low energy consumption, neuromorphic engineering tries to incorporate these superior capabilities into current computing systems.[4–6]

To mimic how the brain works, neuromorphic systems make use of spiking neural networks (SNN). These networks generate and transmit information by means of electrical pulses (spikes) between their different components, achieving asynchronous, and distributed operational capabilities. Since all information within SNNs relies on generating and transmitting spikes, no processing is

D. Casanueva-Morato, A. Ayuso-Martinez, J. P. Dominguez-Morales, G. Jimenez-Moreno
Robotics and Technology of Computers Lab.
Universidad de Sevilla
Sevilla, Spain
E-mail: dcasanueva@us.es

D. Casanueva-Morato, A. Ayuso-Martinez, J. P. Dominguez-Morales, G. Jimenez-Moreno
Escuela Técnica Superior de Ingeniería Informática (ETSII)
Universidad de Sevilla
Sevilla 41012, Spain

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202400282.

**DOI: 10.1002/aisy.202400282**

D. Casanueva-Morato, A. Ayuso-Martinez, J. P. Dominguez-Morales, G. Jimenez-Moreno
Escuela Politécnica Superior (EPS)
Universidad de Sevilla
Sevilla 41011, Spain

G. Indiveri
Institute of Neuroinformatics
University of Zurich and ETH Zurich
Zurich 8057, Switzerland

J. P. Dominguez-Morales, G. Jimenez-Moreno
Smart Computer Systems Research and Engineering Lab (SCORE)
Research Institute of Computer Engineering (I3US)
Universidad de Sevilla
Sevilla, Spain

performed when no spikes are generated. Due to this, SNNs are characterized by lower energy consumption,[7] greater resistance to noise,[8–10] and higher real-time efficiency compared to traditional systems.[11,12]

From a biological point of view, the brain is closer to an analog computer.[13] Analog computing, by exploiting the physical variables (continuous time evolution of voltages, currents, and conductance changes), allows the dynamics of neurons and synapses to be emulated.[4,14,15] When moving to physical implementations, analog neuromorphic hardware has better performance, energy efficiency, and scalability than its digital counterparts.[8,13,15,16] On the contrary, the main disadvantage of neural analog systems is the noisy nature of analog circuits themselves and mismatch.[8,13] However, the brain itself directly and efficiently processes signals that are noisy.[17] Therefore, the design of neuromorphic systems would benefit from the use of analog computation coupled with a noise-tolerant designs.[8]

In any computing system, memory plays an important role. However, within neuromorphic engineering, memory systems are in early stages of development. Of all the regions of the brain involved in memory, the hippocampus is the most prominent. On the one hand, it is a short-term memory capable of storing large amounts of information from various cortical sources in the form of memories by associating the different components that make up this input information. Moreover, from a memory fragment, it can sequentially recall the complete memory. On the other hand, due to the exploitation of its characteristic as a self-associative sequential memory and the functional diversity of the neurons that compose it, it is capable of performing a certain amount of computation within the memory itself. Consequently, the hippocampus is involved in navigation, making it possible to map the environment and retrieve trajectories to goal positions.[18,19]

The hippocampus achieves these characteristics due to its structure and the existence of special neurons called place cells (PC). On the one hand, the structure of the hippocampus is composed of three main brain regions (**Figure 1**a): dentate gyrus (DG), cornu ammonis 3 (CA3), and cornu ammonis 1 (CA1). The spiking input information reaches DG, which, due to its competitive network structure, increases the dimensionality, i.e., it carries out a dispersion of the input information. This information then reaches CA3, which consists of a recurrent collateral network (Figure 1b). Thanks to oscillations of the input activity, this region is responsible for learning and storing the information in the collateral connections themselves and, at the same time, recalling it completely from a fragment. Finally, the memory arrives at CA1, where the original format of the input information is recovered before leaving the hippocampus.[18,19] Furthermore, the main input and output pathway of the hippocampus is the entorhinal cortex (EC), which allows a part of the output information to return to the input, allowing the hippocampus to act as a sequential memory.

On the other hand, PC neurons are activated when the subject is within a certain position in the environment and, therefore, enable the creation of a spike representation of the environment in the memory itself that can be recalled at any moment.[18,20]

Bio-inspired neuromorphic models of the hippocampus and their application to memory systems or robotic navigation systems have been previously addressed in the literature. From the point of view of memory models, several works propose spike-based sequential memory models bio-inspired by the hippocampus.[21,22] Other articles propose spike-based bio-inspired hippocampal memory models but not sequential models.[23–26] If the spectrum is broadened, different studies[27–31] present bio-inspired hippocampal memory models but not purely spike-based, or[32–35] spike-based but not bio-inspired memory models are proposed.

From the point of view of the spiking systems involved in navigation, Oess et al.[36] proposed a spike-based navigation system where the memory system is static and bio-inspired by the hippocampus. Different studies, such as refs. [37–41], present models focusing on bio-inspired environment mapping based on the hippocampal PC neurons, and others, such as refs. [42–45], propose path-planning techniques.

In the brain, the robustness of the system and its tolerance to noise are fundamental characteristics. Moreover, computational systems often have to deal with noise, either external noise that adds to the signal to be analyzed or internal noise produced by the technology itself (an intrinsic feature of analog computing). Despite this, in neuromorphic engineering, hippocampal

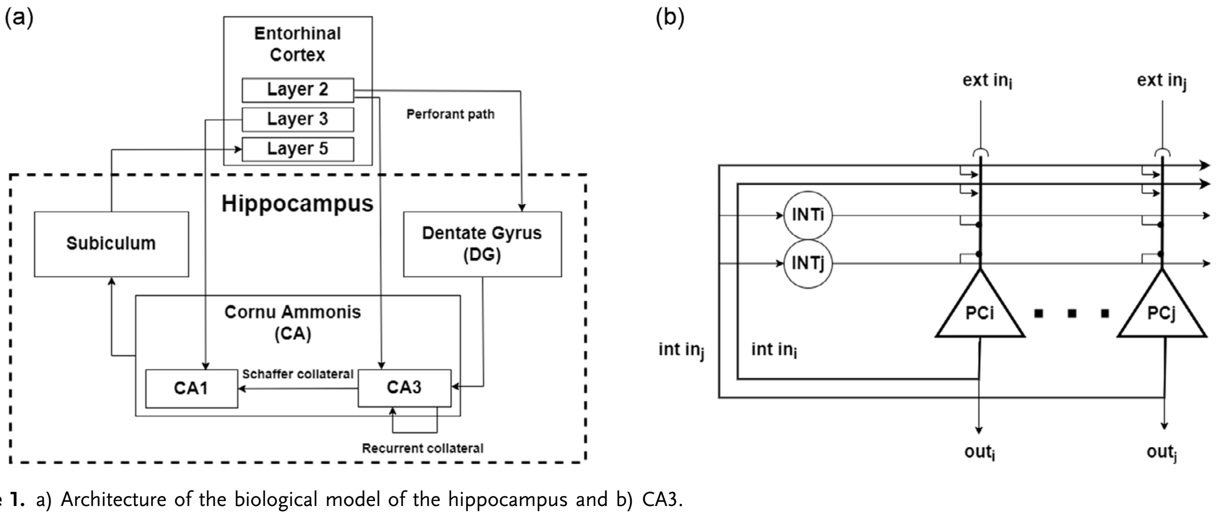


**Figure 1.** a) Architecture of the biological model of the hippocampus and b) CA3.

bio-inspired works dealing with memory systems and robotic navigation systems are scarce, have limitations, and assume ideal scenarios, i.e., they do not analyze the behavior of the proposed system under noisy conditions. In view of this, this article proposes an analog spike-based sequential memory system bio-inspired by the hippocampus. This system is applied to robotic navigation to sequentially recall the trajectory to a goal position within a known environment. Finally, the noise tolerance of the proposed system is analyzed through detailed experimentation, in which the system is subjected to variable noise inputs.

The main contributions of this work include the following: 1) A spike-based bio-inspired hippocampal analog sequential memory model capable of learning, recalling from a fragment, and forgetting sequences of memories. In addition, it can work with both orthogonal and nonorthogonal patterns. 2) Application of the model to robotic navigation. The system is able to learn the different trajectories from any initial position to a goal position within a known environment. In addition, the system is able to recall sequentially, from each initial position, the different positions of the trajectory toward the goal position. 3) A detailed analysis of the system was carried out, proving the high robustness and noise tolerance of the system in this type of application. 4) This is an implementation of a spike-based analog bio-inspired hippocampal sequential memory model on a special-purpose analog hardware platform for SNN systems. 5) The source code is publicly available, together with the documentation, including all the necessary details regarding the SNN architectures.

The rest of the article is structured as follows: Section 2 presents the materials used in this work. In Section 3, the proposed model is detailed. The experiments performed to evaluate the functionality, performance, robustness, and noise tolerance of the proposed model are explained in Section 4, along with the results obtained. Then, in Section 5, the results of the experiments are discussed. Finally, the conclusions of the article are presented in Section 6.

## 2. Materials

### 2.1. SNNs

Hardware implementations of neuromorphic systems typically belong to the third generation of neural networks, known as SNNs.[46] These networks consist of clusters of neurons that dynamically process incoming signals and generate action potentials (spikes) when the combined inputs surpass a certain threshold. They transmit their spikes to their target neurons instantaneously, via synapses. SNNs offer significant computational efficiency by transmitting spikes only when they occur, and they can be programmed to perform complex computations by utilizing sparse activity in both spatial and temporal dimensions.[47]

Various computational models can be used to implement each component of the SNN, aiming to approximate the biological behavior observed in nature. Among these models, the Leaky Integrate-and-Fire (LIF) model has gained significant popularity.[48,49] In the LIF model, the neuron's membrane potential is driven by the sum of input currents, which are generated by synapses stimulated by incoming spikes. If the total input current exceeds the neuron's "leak" current, the membrane potential increases until it reaches a threshold. Conversely, if the total input current is below the leak current, the membrane potential returns to the neuron's resting state. Once the threshold is reached, the neuron generates a brief pulse, known as a spike, and resets its membrane potential. Following the generation of a spike, the neuron remains in a reset state for a specific period of time, known as the refractory period, before it begins integrating its input current again.[50]

The characteristics that define connections between synapses are direction, delay, and weight. The delay attribute represents the time it takes for a spike to travel from the presynaptic neuron to the postsynaptic neuron. The weight attribute indicates the magnitude of the change in the postsynaptic neuron's membrane potential. Lastly, the direction attribute determines whether the change is positive (for excitatory synapses) or negative (for inhibitory synapses).

The learning rules that govern the process of acquiring and storing information play a crucial role in neural networks. While there has been extensive research on spike-timing-dependent plasticity (STDP) learning rules in SNNs,[51–53] recent studies have discovered spike-based synaptic plasticity mechanisms that take into account additional factors, such as the neuron's membrane potential or its recent firing activity, which have proven to be more effective. For a comprehensive overview of these rules, which are also compatible with neuromorphic hardware, please refer to ref. [54].

The STDP triplet rule, which is an expansion of the plain STDP rule, provides a more precise replication of experimental data obtained from actual synapses.[55] In contrast to the basic STDP rule, which only considers pairs of presynaptic and postsynaptic spikes to calculate changes in synaptic weight (increasing the weight if the postsynaptic spike follows the presynaptic one, and decreasing it otherwise), the STDP triplet rule takes into account additional scenarios. These scenarios include a presynaptic spike followed by a postsynaptic spike and another presynaptic spike, as well as a postsynaptic spike followed by a presynaptic spike and another postsynaptic spike. In the former scenario, the synaptic weight decreases, while in the latter scenario, it increases. This learning rule can accurately reproduce the precise spike-timing behavior of the basic STDP rule at low frequencies of input/output spikes. Furthermore, it can explain and replicate Hebbian-type rules based on rates or correlations in high-frequency regimes.[56]

### 2.2. The DYNAP-SE Chip

While there is a wide range of dedicated digital neuromorphic processors that can implement SNNs,[57–59] the research on mixed-signal analog/digital implementations is still ongoing. Currently, only proof-of-concept prototypes of such implementations are available. In this study, we make use of one of these prototypes called the "Dynamic neuromorphic asynchronous processor - scalable" (DYNAP-SE).[60]

DYNAP-SE is a hardware platform that utilizes a scalable multicore architecture with various memory structures to support dynamic asynchronous event-based processing. This platform combines analog circuits, which implement synaptic and neural dynamics, with digital circuits that handle network connectivity and spike routing among firing neurons. Each DYNAP-SE chip consists of four cores, with each core accommodating 256 neurons. These neurons have a fan-in of 64 synapses and a fan-out of 4000 synapses. The chips were manufactured using 0.18 μm 1P6M CMOS technology and incorporated hierarchical asynchronous routers, as well as distributed integrated asynchronous SRAM and CAM memory cells across the cores. This study utilized a board consisting of 4 DYNAP-SE chips. The circuits within each core, which include parameters such as neuron leakage and refractory period, have the same nominal value. However, due to device mismatch, the actual value of each circuit may differ. The DYNAP-SE circuits have a typical coefficient of variation of around 20%.[8] As a result, there can be a notable difference between the specified values and the actual values of these parameters.

The neuron circuits in the DYNAP-SE implement a model equivalent to the Adaptive-Exponential Integrate and Fire (AdExp-I&F),[61,62] whose parameters can be configured to behave like LIF neurons. Synapses and biophysically realistic synapse dynamics are implemented using the current-mode differential pair integrator (DPI) log-domain filter,[63] which can be configured to give rise to 4 possible synapse types: AMPA (fast, excitatory), NMDA (slow, excitatory), GABA B (subtractive inhibitory), and GABA A (shunting inhibitory).

## 3. Sequential Analog Hippocampus Computational Memory Model

### 3.1. Architecture

This work is based on and extends the model proposed in ref. [26]. The architecture of the spike-based hippocampal bioinspired analog model of a sequential memory is presented in **Figure 2**. The model works with real information encoded in trains of 5 to 20 spikes within time windows of 5 to 10 milliseconds (ms). Its architecture consists of 4 main neural structures: DG, CA3, CA1, and EC, where the novelty with respect to previous work lies in EC.

DG performs the dispersion of the content of the input memory. This dispersion only applies to a fragment of the entire memory. The dispersed part of the memory will be referred to as the cue and the remaining part of the memory that remains unchanged is referred to as the content. Concretely, the maximum possible degree of sparsity is applied, i.e., one-hot encoding.

To achieve this functionality, DG presents a multilayer cascaded filter structure that acts as a competitive Winner-Take-All network.[26] For each layer of the filter, the activity corresponding to the cue of the input memory will be propagated and the activity of a set of combinations of input neurons will be computed to determine which output neuron should be activated. If any of the output neurons of that layer is activated, the activity of this neuron will represent the cue of the sparse memory and will propagate through each of the remaining layers of the filter until it reaches the output. At the same time, this activity will inhibit the remaining output neurons in consecutive layers of the filter to prevent more than one output neuron from being activated at the same time.

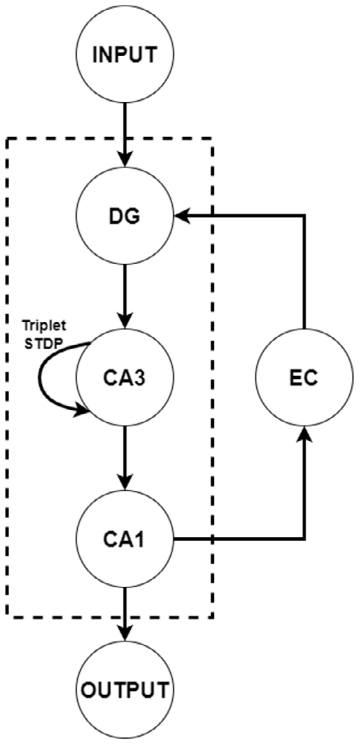


**Figure 2.** Architecture of the sequential analog hippocampus computational memory model proposed. The model divides its architecture into 4 blocks: DG, CA3, CA1, and EC. The word Triplet STDP marks those synapses that exhibit the triplet implementation of the STDP learning mechanism.

CA3 is responsible for the learning, storage, and subsequent recall of individual memories that reach the model. For this purpose, it has a recurrent collateral network structure that can be divided into 2 subpopulations of neurons: the population that receives cue-related activity (CA3cue) and the population that receives content-related activity (CA3cont).[26] CA3cue neurons have lateral all-to-all connections with CA3cont neurons, where the triplet STDP learning mechanism is located. The activation of neurons in CA3 as a consequence of the arrival of a complete memory will result in the learning of the cue-content pair and the storage of this information in the synapses themselves. At the same time, the activation of neurons in CA3 as a consequence of the arrival of a fragment of the memory, corresponding to the cue, will trigger the recall of the complete memory.

CA1 is in charge of recovering the original format of the memory before it exits the model. To attain this, it has a structure similar to DG, but with a single layer.[26] All those DG neurons that were activated by input neurons $i$ will now participate in the activation of output neurons $i$ of CA1. In this way, the activity of the dispersed cue returns to its original format while the activity of the content remains unchanged.

Finally, there is EC. This is a chain of neuron populations that propagate the output activity of a part of the memory back to DG and where the delay of the propagation is regulated by the number of layers in the chain. Specifically, the activity of a subset of neurons belonging to the memory content in CA1 will be propagated and connected to the neurons that will encode the cue at the DG input. This population endows the network with the ability to recall sequences of linked memories.

In a generic way, for a memory of size $M$ in a memory with a capacity of $N$ memories and a delay factor (number of populations to reach a given delay from the base delay of the synapses) of $D$, the first $\lceil \log_2(N+1) \rceil$ neurons would correspond to the cue (*cueSize*) and the remaining $M - cueSize$ would correspond to the content (*contSize*). Inside the content neurons, the activity of the last *cueSize* neurons can be distinguished as content identifying the next memory in the sequence and the first $M - 2 * cueSize$ neurons as content with information belonging to the current memory. Taking these variables into account, the model would have a consumption of $3 * M + 2 * N + cueSize^2 * (cueSize - 2) + D * cueSize$ neurons, $4 * M + 2 * N + D + cueSize * (cueSize^3 - 2 * cueSize^2 + 3 * cueSize - 6 + D)$ static synapses, and $N * contSize$ dynamic synapses with the triplet STDP learning mechanism.

## 3.2. Operating Principle

The spatio-temporal coding of the model's spiking activity is based on the concept that the activation of neurons from the same population at a particular time instant or window is associated with the same memory, while the activation of these neurons but at different time instants or windows is linked to different memories.

The proposed model is capable of learning, recalling from a fragment, and forgetting individual memories, as well as sequential recall of memories.

Learning starts with the input of a complete memory through DG. The part corresponding to the cue will be dispersed, while the content will pass through unchanged. This activity will reach the CA3cue and CA3cont subpopulations of CA3, respectively, triggering the activation of the triplet STDP learning mechanism and, thus, the learning and storage of the memory. For the triplet STDP learning mechanism to work with spike trains, it was configured in such a way that each pair and triplet of spikes trigger small synaptic variations.[26] Specifically, the input of 3 separate spike trains is necessary to ensure the correct learning of the complete memory which, after leaving CA3, will reach CA1, where it will recover the original format and leave the network. At the same time, this configuration prevents independent spikes produced by punctual noise from causing enough synaptic change to carry out a correct learning. Taking this into account, for each spike train, a temporal separation of 100 ms is required to prevent the learning mechanism from mixing the synaptic weight variations of one train with those of the next, with the learning operation having a duration of 300 ms.

The recall operation for a memory starts with the input of neural activity corresponding to only the cue of the memory to be recalled. After being dispersed by DG, it will reach the CA3cue subpopulation in CA3, activating the corresponding neuron. This activation will propagate to CA3cont, activating the subset of content neurons that were associated with the memory by previous learning. In this way, from a fragment of the memory, the complete memory is obtained, which will pass through CA1 to recover its original format before leaving the network. The time it takes for the network to recall the complete memory from the time its cue is introduced until the complete memory is obtained is 25 ms.

This would be the case of recall operations for a single memory. If the activity of the memory that is passed back to DG, through EC, contains some kind of information concerning another memory (namely, the cue), this information would enter the network again to start its recall. In short, thanks to EC in conjunction with the use of a subset of neurons from the content of one memory to contain the information concerning the cue of the next memory in a sequence, sequential recall is achieved. The propagation delay activity of the neurons that return the information to the input to achieve this sequentiality was regulated to allow time for the input spike train to be processed by the network before starting the next memory in the sequence.

Finally, there is the forgetting memory operation. This operation does not occur explicitly in the network, but occurs indirectly by attempting to learn a memory whose cue is common to another previously learned memory. At the start of the learning operation, the new memory will pass through DG which, while dispersing the activity of the cue, creates a time shift in the spatial encoding of the memory. That is, the content of the memory will leave DG and thus enter CA3 before the cue. This cue, being common to another previously learned memory, will trigger the recall of this previous memory. In addition, CA3cont neurons are configured to reduce the frequency of the input spike train.

Putting these factors together, the following sequence of neuron activation occurs. First, the neurons encoding the content of the new memory are activated, followed by the activation of the neurons encoding the cue, and finally, the activation of the neurons encoding the content of the old memory. This sequentiality is exploited by the learning mechanism and results in a decrease in the weight of the synapses storing the old memory, causing it to be forgotten, and an increase in the weight of the synapses storing the new memory, causing it to be learned. This operation requires the same execution time as a normal learning operation, i.e., 300 ms.

## 3.3. Application to Sequential Trajectory Recall

The sequential hippocampus memory model was applied to robotic navigation. The purpose of this application is for the memory model to be able to learn and recall different trajectories to a goal position within a grid environment. This information must be sufficient, in order for the robot to be able to reach a goal position from any position within the environment.

To this end, the content of the memory will be interpreted as a map of the environment where each position contains information about the next position the robot has to move to to reach the goal position. Therefore, each memory will consist of a unique position in the environment as the cue and the next neighboring position to move to as the content. In this case, the content of the memory will be constituted only by the information of the next memory. After learning the complete map information, by starting a recall operation on an initial position in the environment, the hippocampal memory model can return the sequence of positions that the robot has to pass through to reach the goal position. In other words, the hippocampal model is able to tell the robot, for each position in the environment, which trajectory it must follow to reach the goal position.

## 4. Experimentation and Results

The proposed bio-inspired sequential hippocampus memory model was implemented on the DYNAP-SE hardware platform. This hardware implementation of the model presents a capacity to learn and store up to 15 different memories at the same time ($N$), where each memory is defined by the activity of 8 neurons ($M$) and with a delay factor of 24 ($D$). Of the 8 neurons that define memory, the first 4 represent the cue and the remaining 4 the content. For the specific experiments carried out, we focused on testing the sequentiality of the memory. Therefore, the 4 content neurons are used to identify the next memory to be recalled in the sequence, i.e., the next cue in the sequence. For these characteristics, taking into account the equations given in Section 3, the network defining the model will be constituted by a total of 182 neurons (59 from DG, 19 from CA3, 8 from CA1, and 96 from EC), 334 static excitatory and inhibitory synapses (24 from IN-DG, 144 from DG-DG, 19 from DG-CA3, 19 from CA3-CA1, 8 from CA1-OUT, and 120 from EC-EC, CA1-EC, and EC-DG), and 60 dynamic synapses with the triplet STDP learning mechanism (from CA3cue-CA3cont). All inhibitory synapses used are GABA B type and, although this particular implementation was used, the model was also tested for other network sizes of smaller and larger capacity in terms of both number of memories and memory size.

Regarding the hardware implementation of the model, a set of experiments was developed to verify its correct functioning. These experiments consisted in stimulating the model by connecting its input to a neural activity generator (contained in the hardware platform itself) to observe its behavior in response to the input of information in the form of memories. Specifically, all experiments were carried out in the context of applying the model to sequential trajectory recall with and without noise. Each memory consists of the position it represents in the grid environment as the cue and the next position in the sequence as the content. Experiments to verify the basic functionalities can be found in ref. [26].

In these experiments, the grid map of the known environment to learn and recall trajectories is presented in **Figure 3**. It has a total of 16 positions. Position 2 is the goal position and position 16 is not accessible. None of these special positions has any additional information. The remaining 14 positions mark the next position to reach the goal position. The grid map supports 8 positions adjacent to the current one, although, for the initial case, only the 4 main positions (north, south, east, and west) are used. Given an initial position, it is guaranteed that there is a sequence of positions that allows the goal position to be reached. This sequence of positions is obtained by following the arrows indicated in each position. For example, starting from position 11, the next position would be 7, 6, and 2, respectively.

In addition, each experiment includes a raster plot that gives a summary of the network's spiking behavior from the input activity to the network's output. The $x$-axis represents the progression of time in milliseconds, while the $y$-axis represents each neuron in the network, identified by its population and internal ID. Each point on the plot represents a spike fired by the neuron specified on the $y$-axis at the corresponding time on the $x$-axis.

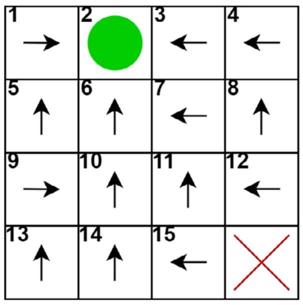


**Figure 3.** Grid map of the known environment over which the robot will navigate. For each position, the position identifier is marked in the upper left corner and, in the center, the next position through which to advance (marked by an arrow) to reach the goal position (green circle). The red cross marks a position that is not accessible due to the presence of an obstacle.

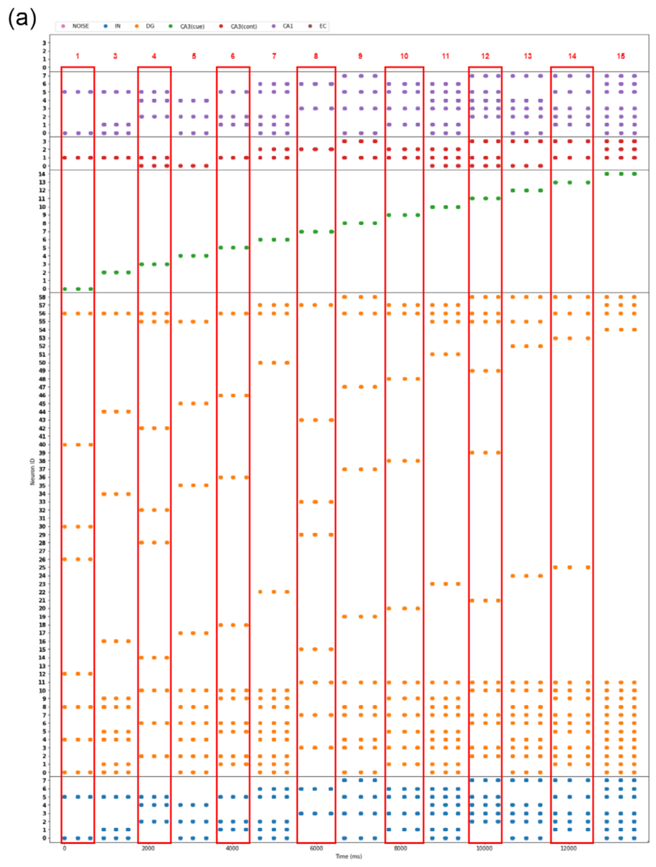
### 4.1. Operational Test: Learning and Sequential Recalling without Noise

The first experiment attempts to demonstrate the sequential operation of the model with the incorporation of EC. To this end, this experiment has a first phase of learning the trajectory map of the known environment, and a second phase of recalling the trajectories from each possible initial position to the goal position.

In the learning phase, a total of 14 learning operations are performed, one for each of the positions with information about the next position in the sequence. This phase begins with the learning of position 1 at ms 0 and ends with the learning of position 15 at ms 13 000. The activity of the network during this phase of the experiment, along with each of these learning operations marked, are presented in **Figure** 4a.

As an example of learning operation, the learning of position 3 was taken. At position 3, the memory is formed by position 3 as cue and position 2 as content, which corresponds to the activation of the input neurons of the network 0 and 1 for cue and 5 for content, considering a spatial binary encoding of these values. This activity is generated in the form of spike trains and fed into the network a total of three times starting at ms 1000. Each time, the cue will pass through DG to be dispersed, leading to the activation of neuron 2 in CA3cue, while the content will pass unchanged through DG activating neuron 1 in CA3cont. At that instant, the triplet STDP learning mechanism in CA3 will be activated, leading to an increase of the weight at those synapses connecting the activated CA3cue and CA3cont neurons and, with it, the learning of memory occurs. The dispersed memory then arrives at CA1 where it regains its original format, i.e., activation of neurons 0, 1, and 5, before leaving the hippocampus at ms 1750. The same is true for the remaining learning operations with their respective encoding of information.

In the recall phase, a total of 14 recall operations are carried out, one for each of the possible positions in which the robot could find itself (ignoring the goal position itself). The activity



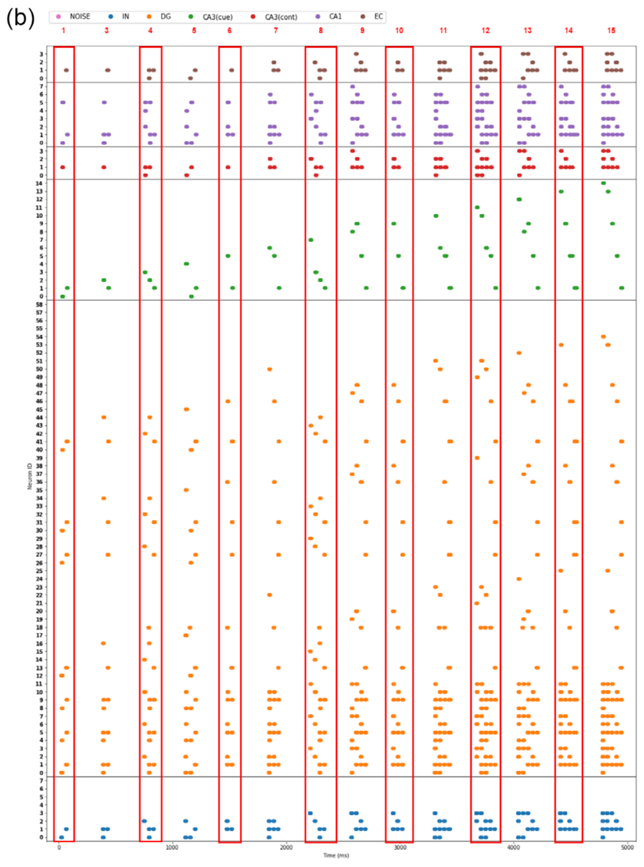


**Figure 4.** a) Raster plot of the spiking activity of the network during the learning of the whole map and b) the recalling of the trajectories from all possible initial positions. Each operation is marked together with the identifier of the map position involved.

of the network during this phase of the experiment, along with each of these recall operations marked, can be observed in Figure 4b.

As an example of recall operation, recall of position 9 was taken. To start the recall operation for position 9, the network receives the activation of input neurons 0 and 3 as a cue via spike trains at ms 1500. After passing through DG, the cue is dispersed, leading to the activation of neuron 8 in CA3cue. The activity train generated by this neuron in CA3cue is transmitted to CA3cont where it will activate those neurons which it was associated with during previous learning (neurons 1 and 3 of CA3cont). These neurons represent position 10, i.e., the next position to which the robot must move to reach its goal position. After passing through CA1 and recovering its original format, the memory will leave the network. At the same time, the activity of these neurons encoding position 10 will return to the network as the cue of a memory through EC (neurons 1 and 3 of IN), triggering the start of a new recall operation immediately.

This second recall operation is performed automatically and, as a result, the next position to 10 (neuron 9 of CA3cue) in the sequence is position 6 (neurons 1 and 2 of CA3cont). This activity will return to the network again through EC as the cue of another memory (neurons 1 and 2 of IN) and will start a new recall operation. In this third recall operation, it is obtained that, if the robot is at position 6 (neuron 5 of CA3cue), the next position toward the goal is position 2 (neuron 1 of CA3cont). Finally, this activity returns to the network input (neuron 1 of IN) to start the fourth and last recall operation for this trajectory. As the current position on the map corresponds to the goal position, there is no next position, i.e., the sequence of memories and recall operations end. This same sequential process occurs for the remaining recall operations, each with its own path length to the goal position and, thus, its own memory sequences.

### 4.2. Noise Analysis Stress Test

This experiment has the same basis as the previous experiment, learning the information from the map and recalling the trajectories from each possible position in the environment. Unlike the previous experiment, it introduces noise at the input of the network to analyze the behavior of the model in the presence of external noise sources. To generate the noise, each input neuron of the network is connected to a Poisson generator available on the DYNAP-SE1 hardware platform.[60,64]

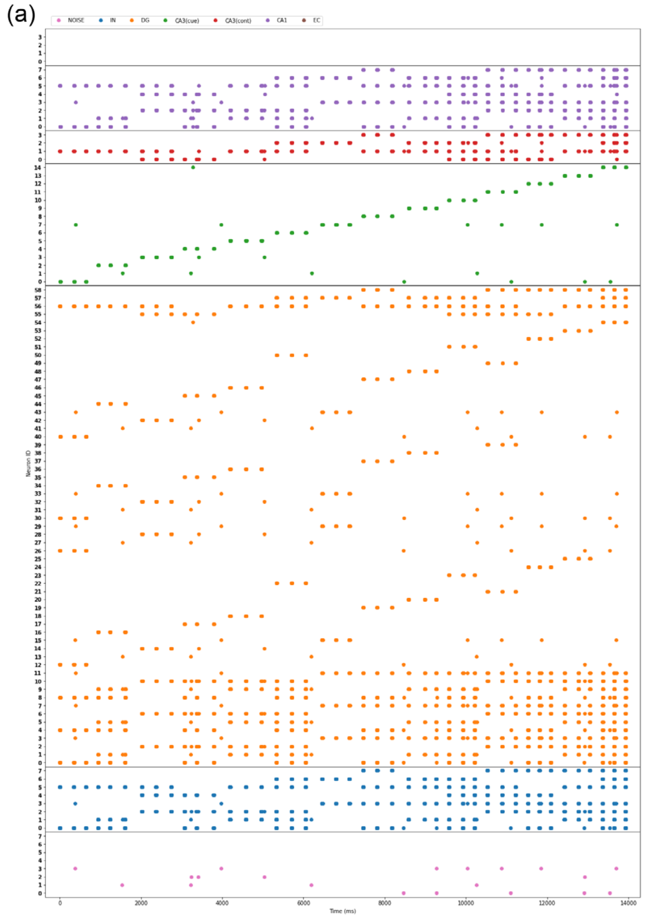
For the presented implementation, a total of 8 Poisson generators are used. Each Poisson generator is able to generate spikes following a Poisson distribution.[65] Thus, given a period, the Poisson generator will generate a spike train in which each spike will be shifted by a random probabilistic error within its period.[64,65] In other words, it will generate a spike at a random instant within each time interval defined by its period. Specifically, the period of operation of the generators is defined by their frequency.

This experimentation considers different variations or test cases of noise application at the network input: 1) The amount of noise applied. This quantity is defined by the frequency in Hz of each of the Poisson generators. The values explored are 0.5, 1, 2, 3, 4, and 10 Hz. 2) The phase of the process that is affected by the noise. A distinction is made between applying noise only to the learning phase, only to the recall phase, and to both phases. 3) The memory fragment which the noise is applied to. A distinction is made between applying the noise only to the cue of the memory, only to the content of the memory, and to the whole memory.

Each of these cases covered in the experimentation was repeated a total of 5 times for each of the 14 possible initial positions of the map. This leaves a total of 756 contemplated cases repeated 5 times or 3780 tests based on the introduction of noise carried out on the model. In addition, for each test, a learning phase and a recall phase were carried out. Due to the large number of test cases in the experiment, there are a large number of figures reflecting the behavior and outcome of the network during these tests. Therefore, the most significant figures are shown, and the remaining figures are added as additional material and are also available in the repository of this work (see Section 6).

**Figure 5** shows the spiking activity of the network as a result of the first test case of the experiment, i.e., the application of 0.5 Hz Poisson noise to the input only during the learning phase and only to the cue of the memory. As the cue of the memory is formed by the activity of 4 neurons, only 4 Poisson generators are used at the input of the network. The noise introduced into the network by these generators (pink dots in Figure 5a) is mixed with the input activity and propagates through the network. It is necessary to look at the output of the network during the recall phase, Figure 5b, to see what effect this noise has had during the learning process. As this is one of the cases with the least amount of noise introduced into the network, all operations were performed correctly except for one, the recall of the trajectory starting at position 8. In the test without noise, the trajectory should start at position 8 and progress through positions 4, 3, and 2, whereas, in this iteration of the test case, the trajectory is defined by positions 8, 15, 14, 10, 6, and 2.

$$SNR(dB) = 10 * \log_{10}\left(\frac{f_{\text{input}}}{f_{\text{noise}}}\right) \tag{1}$$



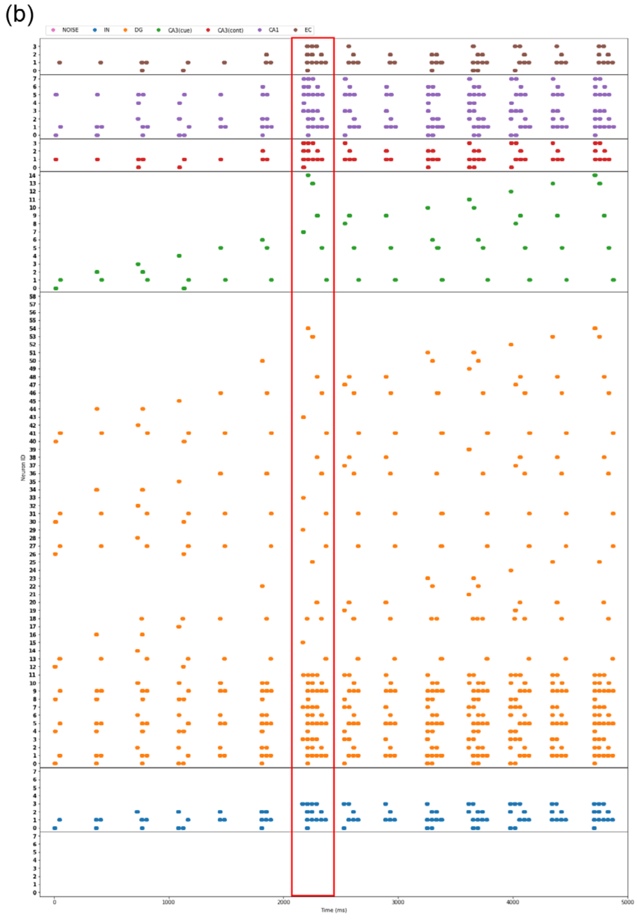


**Figure 5.** a) Raster plot of the spiking activity of the network during the noise analysis stress test for the learning of the whole map and b) the recalling of the trajectories from all possible initial positions. In this test case, a Poisson noise of 0.5 Hz for each generator is introduced into the network only in the learning phase and only applied to the cue of the memory. In (b), a recall operation affected by noise during learning is marked.

In this work, two types of noise can be distinguished: the "synthetic" noise introduced artificially into the model to show the performance of the network in the presence of an external noise source, and the noise introduced by the network and the platform itself derived from this input noise. To measure the impact of the input noise on the network, the signal-to-noise ratio (SNR) was calculated from Equation (1). This indicator allows analyzing the amount of noise with respect to the input information from the frequency of both. On the one hand, $f_{\text{input}}$ is the input frequency of useful information external to the network, it is defined by the input frequency of the spike trains of the different operations multiplied by the average of spikes that define these spike trains. For the case of the implementation used, the spike trains present, on average, 10 spikes and have a frequency of 7 Hz, giving a $f_{\text{input}}$ equal to 70 hz. On the other hand, $f_{\text{noise}}$ is the external noise input frequency and is defined by the frequency of the Poisson generators times the number of Poisson generators acting. The SNR measured in decibels (dB) for the 5 frequencies explored in the 2 possible cases (depending on which part of the memory is affected) is shown in **Table 1**.

We also considered the internal noise factor (INF) at the internal level of the network, not only at the input, to measure the noise induced by the network and the platform derived from the input noise. This factor is calculated from Equation (2), and is the result of dividing the amount of activity generated by the noise in the network by the amount of activity belonging to the useful information in the network. The average INF of this experimentation as a function of the frequency values of the Poisson noise generators is shown for the learning phase in **Figure 6**a and for the recall phase in Figure 6b.

$$\text{INF} = \frac{\text{number of spikes of noise in the network}}{\text{number of spikes of useful information in the network}} \tag{2}$$

**Table 1.** SNR of the experiment as a function of the different frequencies of the Poisson noise generator for the set of test cases where the noise is only applied to the cue or only applied to the content of the memory (A) and for the set of test cases where the noise is applied to the complete memory (B).

| Frequency [Hz] | 0.5 | 1 | 2 | 3 | 4 | 10 |
|---|---|---|---|---|---|---|
| SNR (dB) case A | 15.44 | 12.43 | 9.42 | 7.66 | 6.41 | 2.43 |
| SNR (dB) case B | 12.43 | 9.42 | 6.41 | 4.65 | 3.4 | 0 |

To analyze the performance of the network, the hit rate was measured for each of the individual recall operations (recall hit rate) and for each recall of the entire path (path hit rate). In the recall hit rate, a hit is considered if the recall operation gets the next expected position, the one that would be obtained if the network were free of noise. In the path hit rate, it is considered a hit if, after finishing the recall sequence for an initial position, it is able to reach the goal position, regardless of whether it took the expected or an alternative path. The mean and standard deviation of the recall hit rate (**Figure 7**a and **Table 2**) and of the path hit rate (Figure 7b and **Table 3**) were calculated as a function of the frequency of the noise generators and the SNR values for each of the 9 sets of possible test cases. These sets of test cases arise from the combination of the possible phases and the possible fragments of the memory to which the input noise can be applied.

Finally, to check the impact of noise on the network at the operation level, the average number of recall operations as a function of the frequency of the noise generators was obtained for each of the 9 sets of possible test cases. The results of these

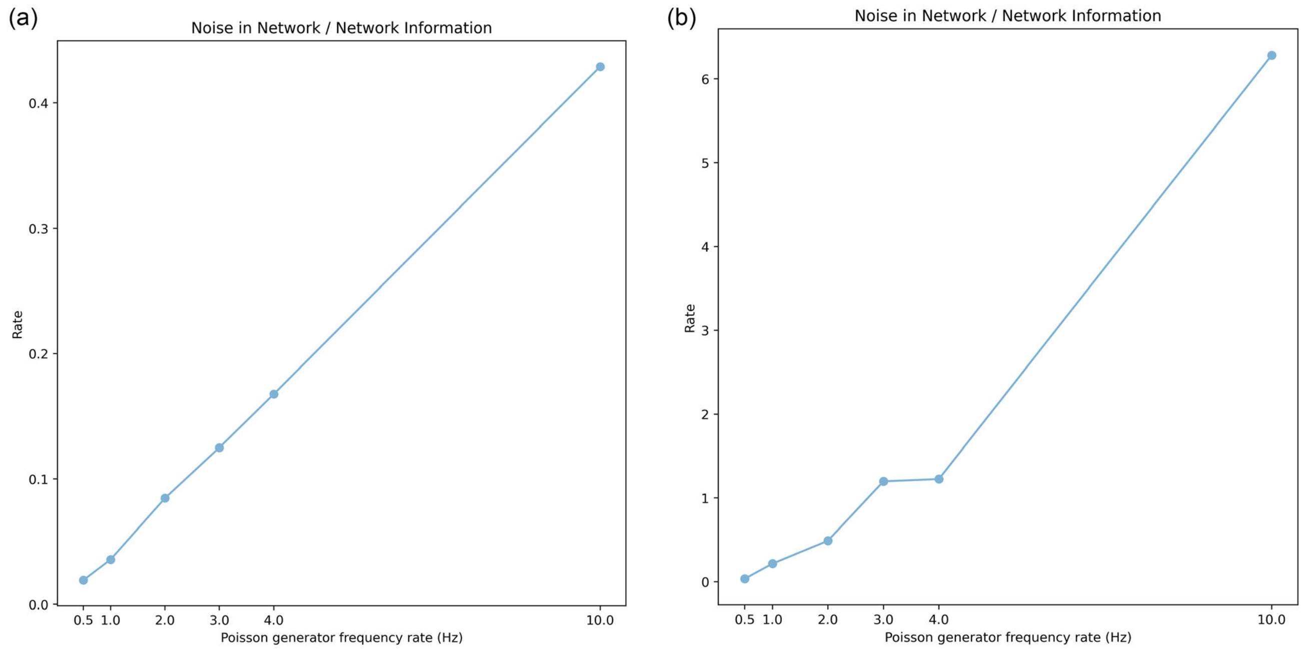


**Figure 6.** a) Mean rate of noise activity in the network versus useful information activity in the network during the noise analysis stress test for the different frequencies of the Poisson noise generators for the learning phase and b) the recalling phase.

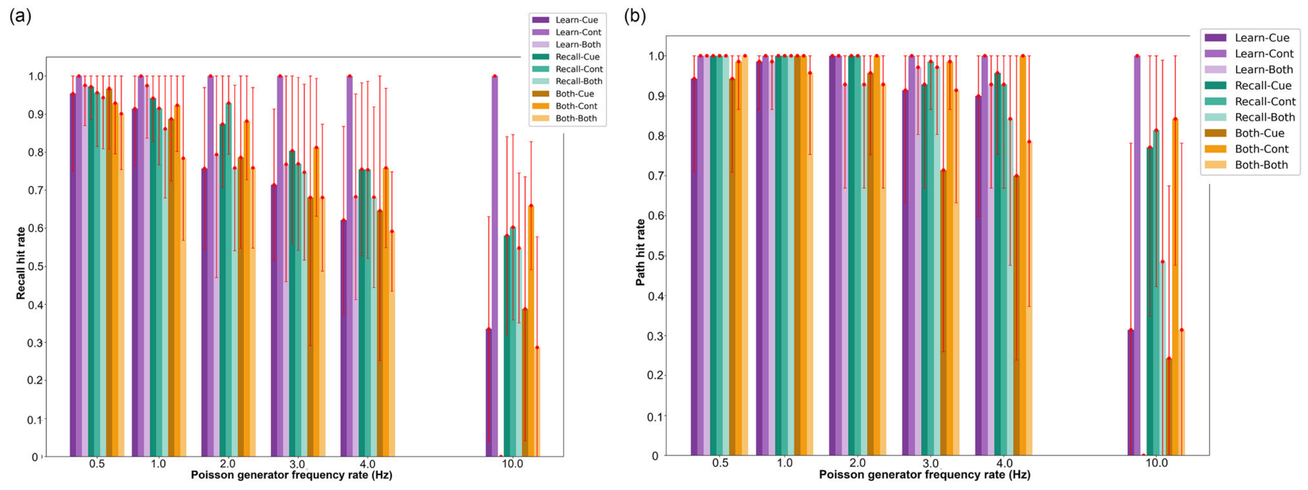


**Figure 7.** a) Mean and standard deviation of the hit rate of the recall operation and of the path recall b) for the different frequencies of the Poisson noise generator. For each frequency, the color of each bar identifies the specific test case of the experiment. The name of the test case identifies in its first part the phase affected by the noise and in the second part the memory fragment it affects.

**Table 2.** Mean and standard deviation of the recall hit rate for different SNR values. The name of the test case identifies in its first part the phase affected by the noise and in the second part the memory fragment it affects. For each SNR, its value in dB is determined for test case A, and test case B. Test case A is that set of test cases where the noise is only applied to the cue or only applied to the contents of the memory, and test case B is that set of test cases where the noise is applied to the entire memory.

| SNR (dB) case A/case B | Experiment case | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Learn-cue (A) | Learn-cont (A) | Learn-both (B) | Recall-cue (A) | Recall-cont (A) | Recall-both (B) | Both-cue (A) | Both-cont (A) | Both-both (B) |
| No noise | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) |
| 15.44/12.43 | 0.954 (±0.204) | 1 (±0) | 0.975 (±0.105) | 0.972 (±0.085) | 0.957 (±0.141) | 0.944 (±0.135) | 0.967 (±0.158) | 0.929 (±0.134) | 0.901 (±0.146) |
| 12.43/9.42 | 0.915 (±0.151) | 1 (±0) | 0.974 (±0.137) | 0.942 (±0.114) | 0.916 (±0.148) | 0.861 (±0.182) | 0.888 (±0.163) | 0.923 (±0.121) | 0.784 (±0.216) |
| 9.42/6.41 | 0.757 (±0.212) | 1 (±0) | 0.794 (±0.323) | 0.874 (±0.168) | 0.929 (±0.134) | 0.759 (±0.217) | 0.786 (±0.238) | 0.882 (±0.154) | 0.759 (±0.210) |
| 7.66/4.65 | 0.714 (±0.199) | 1 (±0) | 0.768 (±0.308) | 0.804 (±0.245) | 0.770 (±0.226) | 0.747 (±0.230) | 0.681 (±0.389) | 0.813 (±0.181) | 0.681 (±0.193) |
| 6.41/3.4 | 0.621 (±0.246) | 1 (±0) | 0.682 (±0.270) | 0.755 (±0.227) | 0.754 (±0.232) | 0.682 (±0.230) | 0.646 (±0.393) | 0.759 (±0.209) | 0.592 (±0.156) |
| 2.43/0 | 0.335 (±0.294) | 1 (±0) | 0 (±0) | 0.581 (±0.260) | 0.603 (±0.243) | 0.548 (±0.197) | 0.388 (±0.346) | 0.660 (±0.167) | 0.287 (±0.290) |

**Table 3.** Mean and standard deviation of the path hit rate for different SNR values. The name of the test case identifies in its first part the phase affected by the noise and in the second part the memory fragment it affects. For each SNR, its value in dB is determined for test case A and test case B. Test case A is that set of test cases where the noise is only applied to the cue or only applied to the contents of the memory, and test case B is that set of test cases where the noise is applied to the entire memory.

| SNR (dB) case A/case B | Experiment case | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Learn-cue (A) | Learn-cont (A) | Learn-both (B) | Recall-cue (A) | Recall-cont (A) | Recall-both (B) | Both-cue (A) | Both-cont (A) | Both-both (B) |
| No noise | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) |
| 15.44/12.43 | 0.943 (±0.233) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 0.942 (±0.233) | 0.986 (±0.119) | 1 (±0) |
| 12.43/9.42 | 0.986 (±0.119) | 1 (±0) | 0.986 (±0.119) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 1 (±0) | 0.957 (±0.203) |
| 9.42/6.41 | 1 (±0) | 1 (±0) | 0.929 (±0.259) | 1 (±0) | 1 (±0) | 0.929 (±0.259) | 0.957 (±0.203) | 1 (±0) | 0.929 (±0.259) |
| 7.66/4.65 | 0.914 (±0.281) | 1 (±0) | 0.971 (±0.167) | 0.928 (±0.259) | 0.986 (±0.119) | 0.971 (±0.167) | 0.714 (±0.455) | 0.986 (±0.119) | 0.914 (±0.281) |
| 6.41/3.4 | 0.900 (±0.302) | 1 (±0) | 0.929 (±0.259) | 0.957 (±0.203) | 0.929 (±0.259) | 0.842 (±0.366) | 0.700 (±0.461) | 1 (±0) | 0.786 (±0.413) |
| 2.43/0 | 0.314 (±0.467) | 1 (±0) | 0 (±0) | 0.771 (±0.422) | 0.814 (±0.391) | 0.485 (±0.503) | 0.243 (±0.431) | 0.843 (±0.366) | 0.314 (±0.467) |

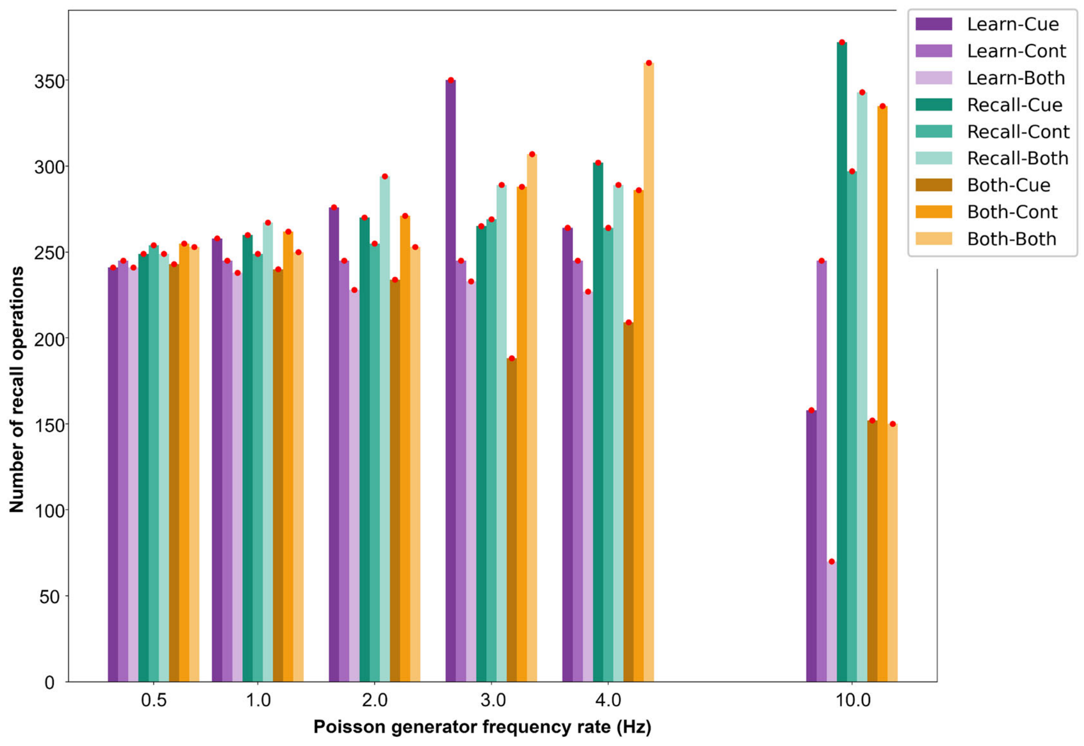


**Figure 8.** Number of recall operations during the noise analysis stress test for the different frequencies of the Poisson noise generator. For each frequency, the color of each bar identifies the specific test case of the experiment. The name of the test case identifies in its first part the phase affected by the noise and in the second part the memory fragment it affects.

measurements are shown in **Figure 8**. Complementarily, to check how the impact of noise on the number of operations translates into the length of the trajectories, the length of the recalled trajectories was measured with respect to the expected ones. **Figure 9** shows, as a function of the frequency of the noise generators, for each expected trajectory length, the actual length measured.

## 5. Discussion

The first experiment carried out in Section 4 has a twofold objective. On the one hand, it allowed demonstrating the correct functioning of the hippocampus model as a spike-based analog sequential memory. On the other hand, it shows the applicability of the model to robotic navigation for learning and recalling trajectories by sequences of positions through a known environment. Each recall operation begins with the generation and introduction into the network of the cue that identifies the position from which the robot wants to start. In response, the network will recall that complete memory, i.e., the next position to which the robot should move and, at the same time, this information is used by the network itself to continue the sequence of memories until the goal position is reached. Moreover, thanks to the dynamism of the memory design (ability to learn and forget), the trajectory map could be modified during the robot's own path in real time.

During the design of the model, noise tolerance and robustness were two of the main features to be taken into account. In DG, a cascade filter is used to generate a consistent and unique output for each possible combination of input activity even with the mismatch introduced by the hardware platform and the intrinsic noise of the analog computation. After passing through this structure, both noise sources are corrected. In CA3, the triplet STDP learning mechanism was configured to ensure that the synaptic changes produced by each pair of spikes are small, requiring several continuous spike trains over time to achieve complete learning. This configuration of the learning mechanism, together with a spike-train-based network operation, allows the effect of punctual input noise to be greatly reduced. Finally, there is the use of a spatial representation with spike trains distributed within the time window as a consequence of the time lag introduced by DG. This last property of the system, added to the configuration of the learning mechanism, allows us to obtain learning with the possibility of forgetting memories and, at the same time, allows the small synaptic variations produced by the spike noise to be eliminated if it is not continuous in time.

Due to this design, the second experiment was focused on the analysis of the robustness of the model when applied to robotic navigation for learning and recalling trajectories. According to the results obtained in Section 4, the model is quite robust when used in this type of application. This robustness is analyzed below in terms of the results obtained.

At the recall operation level, with 0.5 Hz of input noise per generator, the model is able to maintain a hit rate above 90%, around 80% with the standard deviation. These are quite good

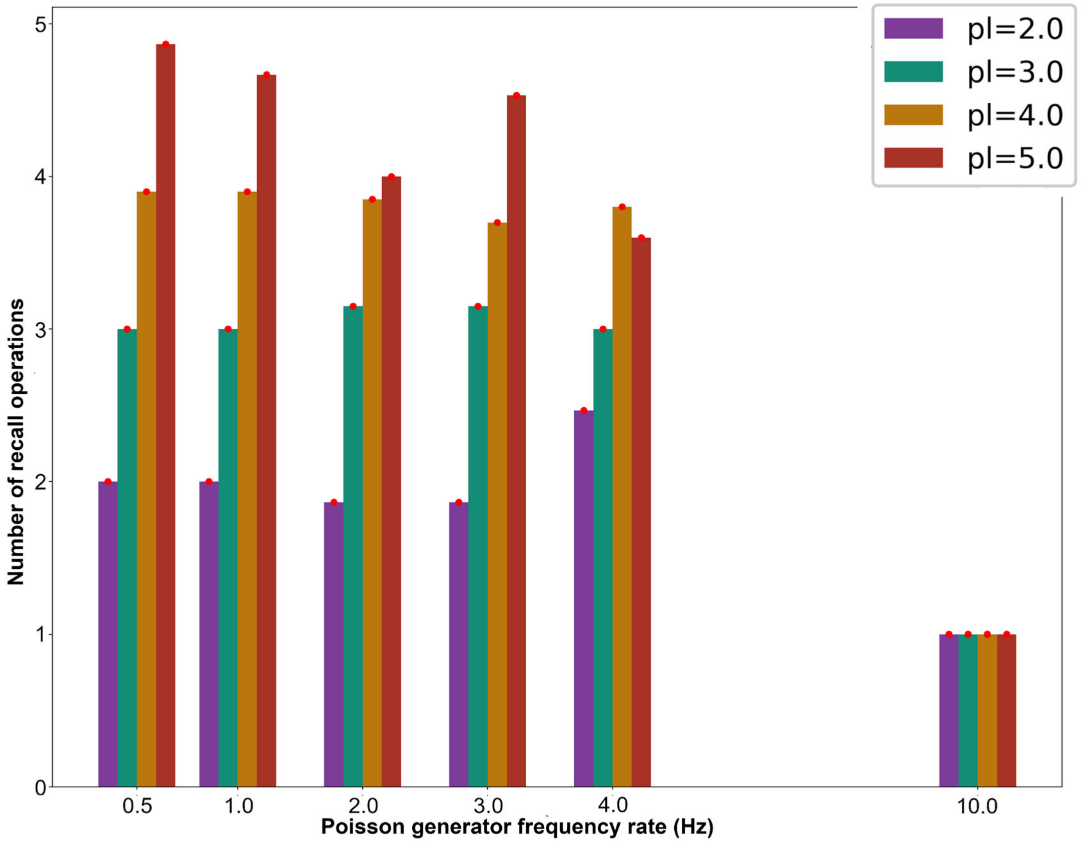


**Figure 9.** Average path length during the noise analysis stress test for each frequency of the Poisson noise generator. The *x*-axis represents the frequency of the input noise generator. The *y*-axis represents the measured path length, i.e., the number of recall operations required to get from the initial position to the goal position during the experiment. The color of the bars represents the expected path length in the absence of noise.

results considering that, in the worst cases, this noise is affecting the learning phase itself and, therefore, memories may be learned incorrectly. As the input noise increases, the recall hit rate suffers a considerable loss, reaching 3 Hz per generator, where, despite having an average recall hit rate of around 70%, it drops below 50% with the deviation in most cases. Even at 2 Hz per generator, although the average is around 80%, many are already approaching the 50% with the deviation. However, at the global path level, the hit rate decrease is much smaller. The model achieves a path hit rate of almost 100% up to 2 Hz of noise per generator with very little deviation, and it is from 3 Hz onward that this accuracy really starts to suffer. Even with noise generators at 4 Hz, the model still achieves an average path hit rate of 80%, above 50% with the standard deviation in most cases.

This means that, despite making a mistake in the recalling of the next position to go to, the model is still able to search for an alternative trajectory that eventually leads to the goal. Generally speaking, it can be said that the network is able to function correctly and reach the goal position from any initial position within the environment up to 3 Hz in the noise generators. At 4 Hz, degradation begins to show but is still functional in most cases. It is from 4 Hz onward that this degradation becomes unsustainable and the network gradually starts to malfunction in most cases.

This degradation continues until the network converges to a saturation state at 10 Hz frequency per generator. At this point, the model reaches 2 possible behaviors: it remains in an infinite loop trying to find a route to the goal, but never arrives, or it converges to a position from which it is not able to advance. On the one hand, the first case is related to noise during learning. Consequently, the learned map does not correspond to the real one and, therefore, the robot ends up taking trajectories with loops that do not lead to the goal. In addition, it may also be caused by the fact that, during the recall phase, even if the next position is correctly recalled, due to the large amount of noise, this new cue is constantly disturbed. On the other hand, the second case is related to forgetting in the model. Each operation carried out by punctual noise spikes leads to a small increase and/or decrease of weight in CA3. If the activity generated by the noise ends up reaching or even exceeding the activity of the useful information, these changes in CA3 weights end up having a balance in favor of forgetting previous memories. Therefore, the saturation of the network as a consequence of the input noise leads to an excess of nonuseful information in the first case and an absence of useful information in the second case.

This described behavior is observable in nature. When a subject is performing a task, if it is in the presence of external noise, it can continue to perform the task, although with certain difficulties. These difficulties may be due to the need for a longer duration to perform the task, due to the difficulty in concentrating on it, or due to mistakes in the different steps involved in the task. When noise is excessive, noise distracts it from the task (too much useful information to recall) or prevents it from thinking about the information needed to perform the task (lack of useful information).

The model is considered to have a good tolerance as it is able to operate correctly even with a random noise input of 3 and 4 Hz frequency per Poisson generator. To understand the amount of noise the network is working with, the SNR is used. For 3 Hz, the network will have an input noise of 7.66 and 4.65 dB for the case of applying noise to a fragment of the memory or to the whole memory, respectively. For 4 Hz, the network will have an input noise of 6.41 and 3.4 dB, respectively. SNR is a standardized measure to determine the amount of input noise relative to the input information. However, it is necessary to provide a quantitative measure of noise at the level of the whole network. For this purpose, the INF is used. This measure indicates the factor of noise activity to useful information activity in the whole network. In the learning phase, the noise activity in the network is equivalent to ≈15% (3 Hz of noise frequency) and 20% (4 Hz of noise frequency) of the activity related to useful information. During the recall phase, these values are ≈120% for both frequencies. The latter value is very high due to the fact that, in the network, before, during, and/or after the desired recall operations, noise can evoke partially sequential recall operations. Therefore, it can be said that the model is tolerant of high noise activity.

Despite this, there is a set of special test cases where the model has shown a higher than normal tolerance to noise, far exceeding all other cases. These are experiments where noise is applied only during the learning phase and only to the content part of the memory. In this set of cases, the network achieves a recall and path hit rate of almost 100% even with a noise frequency of 10 Hz per generator. As a result, experimentation was extended for this set of cases, exploring values from 10 to 150 Hz per generator. The model was able to achieve a hit rate of up to 95% hit rate, dropping to 80% with the standard deviation, for the recall and path hit rate with 75 Hz frequency noise per generator. From 75 Hz onward, the degradation started to worsen greatly, reaching saturation at 100 Hz.

The high tolerance achieved in this set of cases may be due to the behavior of the triplet STDP. Due to this learning mechanism and its configuration indicated in Section 3, it requires very specific conditions, thus, the noise applied to the content of the memory modifies its learning. Specifically, it is necessary that the frequency of the noise generators affecting the content of the memory is sufficiently large. When this occurs, the conditions of number of consecutive spikes and in the exact temporal sequence necessary for this noise to be learned are met. In this case, the recall defined by the noise will eventually replace the useful information and have an impact on the hit rate.

In general, the model is more tolerant to noise when it is applied only to the recall phase than to the learning phase. This is due to the fact that, even if there is no input noise during the recall phase, if the map is not learned correctly, it becomes difficult to recall a trajectory that will lead to the goal. Indirectly, noise in the learning phase is carried over to the recall phase. When the amount of noise is low, the disturbances in the memory to learn are small, thus the network finds alternative trajectories. However, when the noise increases and these perturbations are larger, probabilistically, the number of possible trajectories that end up leading to the goal will be smaller and the possibility of ending up in infinite loops increases. Meanwhile, if the noise is applied only to the recall phase, the map is correctly learned. This means that noise disturbances cause the network to jump or not recall correctly some positions, however, from that new misplaced position; there is still a path to the goal. The problem in the case of applying the noise only to the recall phase is when the noise is so frequent that it causes some positions to be forgotten and/or new incorrect positions to be learned, which also occurs when the noise is applied only to the learning phase.

At the same time, the results for the network when noise is applied to both phases are lower than in the cases where noise is applied to only one phase. However, it should be noted that this noise source was applied twice, once per phase. Thus, the results of the network when the noise is applied to both phases at a given frequency should be compared to the results of the network when the noise is applied to only one phase for twice that frequency. The number of spikes related to noise at the network level in the whole process is twice as high when noise is applied to both phases. Thus, it can be said that the network has a slightly higher tolerance when the noise is applied to both phases than when the noise is applied to a single phase if the noise is measured in number of spikes or slightly lower if the noise is measured in SNR. This may be due to the fact that, despite incorrectly learning the map during learning, it is the noise in the recall phase which enables unexpected position jumps and re-routes the model to a possible unexpected trajectory.

Expanding this analysis as a function of the fragment of the memory that is affected by the noise, the model is much more robust when the noise is applied only to the content and much less robust when it is applied only to the cue. The cue of the memory must pass through several layers of processing within the network and is identified as one of the most sensitive parts of the memory, as the subsequent recall operation depends on it, in contrast to the content, which not only has a short processing path in the network, but is also the element to be recalled by the cue. The longer the processing path that the memory fragment affected by the noise has to travel, the greater the propagation of the noise in the network and thus the greater the memory degeneration.

When these results are compared to those obtained when the noise is applied to the entire memory, it is found to perform somewhere in between. That is, the network is more robust when the noise is applied to the entire memory than when it is applied to the cue only, but it is less robust than when the noise is applied to the content only. It is possible that the higher degradation caused by the noise applied to the cue is compensated by the lower degradation caused by the noise applied to the content. However, the SNR in experiments where the noise is applied to the entire memory at a certain frequency is equivalent to the SNR in experiments where the noise is applied to a particular fragment of the memory at twice that frequency. Therefore, at equal SNR, the tolerance of the network in the case of applying the noise to the whole memory is similar to the tolerance in the case of applying the noise to the content only.

Finally, it is necessary to analyze the impact of noise on the number of operations carried out by the network. It can be observed that the network increases the number of recall operations as the noise increases, although this increase is not significant on average. It can be said that the number of operations does not change much with the introduction of noise, which does not seem to fit with some of the previous conclusions. To understand this behavior, it is necessary to turn to the results about the length of the obtained trajectories with respect to the expected ones.

In general, a network trend can be observed in which short paths are getting longer and long paths are getting shorter. On the one hand, this trend is responsible for the fact that the total number of operations has not grown excessively with the introduction of noise. The number of operations that are reduced by the shortening of the longer paths is compensated by the increase in the number of operations on the shorter paths that are getting long. Despite this, there is still a slight increase in the number of operations due to new unwanted recall operations as a consequence of high amounts of input noise. On the other hand, the results show a tendency to normalize the length of the trajectories to an average of between 2.5 and 3 positions with the arrival of the noise. When the model has trouble recognizing where it is or where to go, it eventually finds a stable trajectory to which it converges regardless of the initial position. This trajectory is usually maintained despite large amounts of noise, as the constant recall of such a sequence of positions reinforces this set of memories in memory. That is, each time a position is recalled, the weight of the synapses that store them is increased.

An exception occurs when the noise is generated at a frequency of 10 Hz. In this case, there are cases where the network shows hardly any activity or where this activity is skyrocketing. This is due to the two possible behaviors taken by the network when it saturates: either it converges to a position forgetting part of the map or it enters infinite loops without being able to reach the goal position.

## 6. Conclusion

In this work, a fully functional analog spike-based sequential memory model bio-inspired by the hippocampus was proposed. This model was not only simulated in software, but it was also implemented with SNNs on the DYNAP-SE hardware platform. Furthermore, it was applied to robotic navigation for learning and recalling trajectories from any position to a goal position within a known grid environment. Through extensive experimentation, the robustness and tolerance of the system to sources of random input noise were analyzed and demonstrated. The model was able to achieve a 90% path hit rate even with an SNR of up to 4.65 dB and 80% with an SNR of up to 3.4 dB. Although a specific implementation of the model was shown, its parametric design allows for higher or lower-capacity memory implementations.

There are several implications of noise input in the network. First, the network is able to maintain its functionality of reaching the goal position almost seamlessly in the presence of random external noise. As this noise increases, it begins to slightly modify the trajectories that the model works with and a degradation in its hit rate is denoted. This degradation increases to the point of saturating the network, at which point it stops working due to an excess of nonuseful information or an absence of useful information. Second, in general, noise has a greater impact on the model when it affects only the learning phase and when it is applied only to the memory cue. Similarly, the model is more robust when noise is applied only to the recall phase and when it is applied only to the content of the memory. However, considering the total amount of noise handled by the network during the experiments, the model exhibits optimal noise tolerance even when noise is applied to both phases and to the entire memory.

As a third implication, with noise input, the model tends to normalize the length of the trajectories to 2.5 positions, shortening long trajectories and lengthening short trajectories. In this way, the total number of memory operations increases only slightly due to incorrect operations initiated by the noise. Similarly, as the amount of noise increases, the model tends to take a certain trajectory as preferred among all those capable of reaching the goal. This implies that, when noise prevents it from knowing the next position, it tends to end up on the trajectory that ensures it will reach the goal position.

Despite all this, the memory model and its application to robotic navigation still have room for improvements and extensions that remain for future work. On the one hand, it would be interesting to explore new mechanisms that enable greater control over the flow of sequential oscillatory activity and thus improve noise tolerance, for example, the addition of an inhibitory mechanism with STDP in EC such that the network is able to learn the goal position and, after reaching that position, is able to reduce all subsequent activity. Once the goal position is reached, EC is inhibited to prevent further oscillation due to noise, at least until the next recall operation is performed. On the other hand, it would be interesting to carry out the exploration of the environment and the definition of the trajectories that lead to the goal in a spike paradigm. In addition, there remains, as a future extension, the use of the system with dynamic environments and obstacles, with all the changes that this work implies.

The source code of the implemented model, the experiments and simulations performed together with all the figures generated, including those of the additional material, is available on an open-source GitHub repository (https://github.com/dancasmor/Robust-analog-sequential-hippocampus-memory-model-for-trajectory-learning-and-recalling).

## Acknowledgements

This work is supported by grant MINDROB (PID2019-105556GB-C33), NASSAI (PDC2023-145841-C33), NEKOR (PID2023-149071NB-C54), and SANEVEC (TED2021-130825B-I00) and funded by MICIU/AEI /10.13039/501100011033, by "ERDF A way of making Europe," and by the European Union NextGenerationEU/PRTR. D.C.-M. was supported by a "Formación de Profesorado Universitario" Scholarship and by "Ayudas complementarias de movilidad" from the Spanish Ministry of Science, Innovation and Universities. D.C.-M. would like to thank Giacomo Indiveri and his group for hosting him during a three-month internship between 1st June 2023 and 31th August 2023, during which the idea of this article was originated and most of the results presented in this work were obtained.

## Conflict of Interest

The authors declare no conflict of interest.

## Data Availability Statement

https://github.com/dancasmor/Robust-analog-sequential-hippocampus-memory-model-for-trajectory-learning-and-recalling.

## Keywords

analog sequential memory, DYNAP-SE, hippocampus model, neuromorphic engineering, robustness analysis, spiking neural networks

Received: April 9, 2024
Revised: August 27, 2024
Published online: September 30, 2024

[1] A. Vanarse, A. Osseiran, A. Rassau, *IEEE Instrum. Meas. Mag.* **2019**, *22*, 4.
[2] S. Soman, S. Jayadeva, M. Suri, *Big Data Anal.* **2016**, *1*, 1.
[3] F. Zenke, S. M. Bohté, C. Clopath, I. M. Comşa, J. Göltz, W. Maass, T. Masquelier, R. Naud, E. O. Neftci, M. A. Petrovici, F. Scherr, D. F. M. Goodman, *Neuron* **2021**, *109*, 571.
[4] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, K. Boahen, *Front. Neurosci.* **2011**, *5*, 73.
[5] Z. Sun, V. Cutsuridis, C. F. Caiafa, J. Solé-Casals, *Cogn. Comput.* **2023**, *15*, 1103.
[6] A. P. Vaz, J. H. Wittig, S. K. Inati, K. A. Zaghloul, *Nat. Commun.* **2023**, *14*, 4723.
[7] S. Kim, S. Park, B. Na, S. Yoon, *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 11 270.
[8] D. Zendrikov, S. Solinas, G. Indiveri, *Neuromorphic Comput. Eng.* **2023**, *3*, 034002.
[9] S. Kundu, M. Pedram, P. A. Beerel, in *Proc. of the IEEE/CVF Inter. Conf. on Computer Vision*, IEEE, Piscataway, NJ **2021**, pp. 5209–5218.
[10] W. Guo, M. E. Fouda, A. M. Eltawil, K. N. Salama, *Front. Neurosci.* **2021**, *15*, 638474.
[11] K. Roy, A. Jaiswal, P. Panda, *Nature* **2019**, *575*, 607.
[12] J. Zhu, T. Zhang, Y. Yang, R. Huang, *Appl. Phys. Rev.* **2020**, *7*, 011312.
[13] D. Ivanov, A. Chezhegov, M. Kiselev, A. Grunin, D. Larionov, *Front. Neurosci.* **2022**, *16*, 1513.
[14] R. Sarpeshkar, *Neural Comput.* **1998**, *10*, 1601.
[15] B. Cramer, S. Billaudelle, S. Kanya, A. Leibfried, A. Grübl, V. Karasenko, C. Pehle, K. Schreiber, Y. Stradmann, J. Weis, J. Schemmel, F. Zenke, *Proc. Natl. Acad. Sci.* **2022**, *119*, e2109194119.
[16] I. Kataeva, S. Ohtsuka, H. Nili, H. Kim, Y. Isobe, K. Yako, D. Strukov, in *2019 IEEE Inter. Symp. on Circuits and Systems (ISCAS)*, IEEE, Piscataway, NJ **2019**, pp. 1–5.
[17] A. Mehonic, A. J. Kenyon, *Nature* **2022**, *604*, 255.
[18] E. T. Rolls, *Brain Computations: What and How*, Oxford University Press, Oxford **2021**.
[19] K. Anand, V. Dhikav, *Ann. Indian Acad. Neurol.* **2012**, *15*, 239.
[20] J. R. Whitlock, R. J. Sutherland, M. P. Witter, M.-B. Moser, E. I. Moser, *Proc. Natl. Acad. Sci.* **2008**, *105*, 14755.
[21] C. H. Tan, E. Y. Cheu, J. Hu, Q. Yu, H. Tang, in *Inter. Conf. on Neural Information Processing*, Springer, Berlin **2011**, pp. 493–500.
[22] C. H. Tan, H. Tang, K. C. Tan, M. Yuan, in *2013 IEEE Conf. on Cybernetics and Intelligent Systems (CIS)*, IEEE, Piscataway, NJ **2013**, pp. 134–139.
[23] D. Casanueva-Morato, A. Ayuso-Martinez, J. P. Dominguez-Morales, A. Jimenez-Fernandez, G. Jimenez-Moreno, in *2022 Inter. Joint Conf. on Neural Networks (IJCNN)*, IEEE, Piscataway, NJ **2022**, pp. 1–9.
[24] D. Casanueva-Morato, A. Ayuso-Martinez, J. P. Dominguez-Morales, A. Jimenez-Fernandez, G. Jimenez-Moreno, A bio-inspired implementation of a sparse-learning spike-based hippocampus memory model, *IEEE Trans. Emerging Topics Comput.* **2024**, pp. 1–16.
[25] R. Shrivastava, P. S. Chauhan, *Comput. Methods Biomech. Biomed. Eng.* **2023**, *1*, 1.
[26] D. Casanueva-Morato, A. Ayuso-Martinez, G. Indiveri, J. P. Dominguez-Morales, G. Jimenez-Moreno, A bio-inspired hardware implementation of an analog spike-based hippocampus memory model, **2024**. https://doi.org/http://dx.doi.org/10.36227/techrxiv.171216721.14143739/v1
[27] M. K. Benna, S. Fusi, *Proc. Natl. Acad. Sci.* **2021**, *118*, e2018422118.
[28] G. Ma, R. Jiang, L. Wang, H. Tang, *Neural Netw.* **2023**, *166*, 174.
[29] Y. Yue, M. Baltes, N. Abujahar, T. Sun, C. D. Smith, T. Bihl, J. Liu, Hybrid spiking neural network fine-tuning for hippocampus segmentation, arXiv preprint arXiv:2302.07328, **2023**.
[30] T. Zhang, Y. Zeng, D. Zhao, L. Wang, Y. Zhao, B. Xu, in *2016 IEEE Inter. Conf. on Systems, Man, and Cybernetics (SMC)*, IEEE, Piscataway, NJ **2016**, pp. 2301–2306.
[31] Y. Zhang, Y. Chen, J. Zhang, X. Luo, M. Zhang, H. Qu, Z. Yi, in *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, Piscataway, NJ **2022**, pp. 1–15.
[32] R. Mueller, A. V. M. Herz, *Phys. Rev. E* **1999**, *59*, 3330.
[33] M. Matsugu, A. L. Yuille, *Neural Netw.* **1994**, *7*, 419.
[34] H. He, Y. Shang, X. Yang, Y. Di, J. Lin, Y. Zhu, W. Zheng, J. Zhao, M. Ji, L. Dong, N. Deng, Y. Lei, Z. Chai, *Front. Neurosci.* **2019**, *13*, 650.
[35] A. Ayuso-Martinez, D. Casanueva-Morato, J. P. Dominguez-Morales, A. Jimenez-Fernandez, G. Jimenez-Moreno, in *IEEE Transactions on Emerging Topics in Computing*, IEEE, Piscataway, NJ **2023**.
[36] T. Oess, J. L. Krichmar, F. Röhrbein, *Front. Neurorobot.* **2017**, *11*, 4.
[37] R. Kreiser, A. Renner, Y. Sandamirskaya, P. Pienroj, in *2018 IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ **2018**, pp. 2159–2166.
[38] R. Kreiser, G. Waibel, N. Armengol, A. Renner, Y. Sandamirskaya, in *2020 IEEE Inter. Conf. on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2020**, pp. 6134–6140.
[39] G. Tang, K. P. Michmizos, in *Proc. of the Inter. Conf. on Neuromorphic Systems* Knoxville, TN, July **2018**, pp. 1–8.
[40] G. Tang, A. Shah, K. P. Michmizos, in *2019 IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ, **2019**, pp. 4176–4181.
[41] H. Nakagawa, K. Tateno, K. Takada, T. Morie, *IEEE Access* **2022**, *10*, 43003.
[42] S. Koziol, S. Brink, J. Hasler, *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2014**, *22*, 2724.
[43] T. Hwu, A. Y. Wang, N. Oros, J. L. Krichmar, *IEEE Trans. Cogn. Devel. Syst.* **2018**, *10*, 126.
[44] S. Koul, T. K. Horiuchi, *IEEE Trans. Circuits Syst. I: Regul. Pap.* **2019**, *66*, 1544.
[45] M. Sakurai, Y. Ueno, M. Kondo, in *2021 IEEE Inter. Conf. on Intelligence and Safety for Robotics (ISR)*, IEEE, Piscataway, NJ **2021**, pp. 209–215.
[46] W. Maass, C. Bishop, *Pulsed Neural Networks*, MIT Press, Cambridge **1998**.
[47] J. L. Lobo, J. Del Ser, A. Bifet, N. Kasabov, *Neural Netw.* **2020**, *121*, 88.
[48] L. F. Abbott, *Brain Res. Bull.* **1999**, *50*, 303.
[49] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, A. Maida, *Neural Netw.* **2019**, *111*, 47.
[50] R. B. Stein, *Biophys. J.* **1965**, *5*, 173.
[51] H. Markram, W. Gerstner, P. J. Sjöström, *Front. Synaptic Neurosci.* **2012**, *4*, 1.

[52] L. F. Abbott, S. B. Nelson, *Nat. Neurosci.* **2000**, *3*, 1178.
[53] J. Sjöström, W. Gerstner, *Spike-Timing Depend. Plast.* **2010**, *35*, 574.
[54] L. Khacef, P. Klein, M. Cartiglia, A. Rubino, G. Indiveri, E. Chicca, *Neuromorphic Comput. Eng.* **2023**, *3*, 042001.
[55] J.-P. Pfister, W. Gerstner, *J. Neurosci.* **2006**, *26*, 9673.
[56] R. Kempter, W. Gerstner, J. L. Van Hemmen, *Phys. Rev. E* **1999**, *59*, 4498.
[57] S. B. Furber, F. Galluppi, S. Temple, L. A. Plana, *Proc. IEEE* **2014**, *102*, 652.
[58] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. Mccoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, H. Wang, *IEEE Micro* **2018**, *38*, 82.
[59] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, D. S. Modha, Alvarez-Icaza, *Science* **2014**, *345*, 668.
[60] S. Moradi, N. Qiao, F. Stefanini, G. Indiveri, *IEEE Trans. Biomed. Circuits Syst.* **2018**, *12*, 106.
[61] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, G. Indiveri, *Front. Neurosci.* **2015**, *9*, 141.
[62] R. Brette, W. Gerstner, *J. Neurophysiol.* **2005**, *94*, 3637.
[63] C. Bartolozzi, G. Indiveri, *Neural Comput.* **2007**, *19*, 2581.
[64] M. Nilsson, Monte carlo optimization of neuromorphic cricket auditory feature detection circuits in the dynap-se processor, **2018**.
[65] D. Heeger, *Poisson Model of Spike Generation, Handout*, Vol. *5*, University of Standford, Stanford, CA **2000**, p. 76.