TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

Design and Implementation of a Biologically Inspired Grid Cell Network

Zibo Zhou, Zhenshan Bing, Alois Knoll

TUM-I2292

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Design and Implementation of a Biologically Inspired Grid Cell Network

Zibo Zhou

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master′s Thesis in Informatics

Design and Implementation of a Biologically Inspired Grid Cell Network

Design und Implementierung eines biologisch inspirierten Gitterzellennetzwerks

Author:	Zibo Zhou
Supervisor:	Prof.Dr.-Ing.habil. Alois Knoll
Advisor:	Dr.rer.nat. Zhenshan Bing
Submission Date:	April 19,2021

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, April 19,2021 Zibo Zhou

Abstract

Grid cells (GCs) in the rat entorhinal cortex display strikingly regular firing responses to the animal's position in 2-D space and have been hypothesized to form the neural substrate for dead-reckoning. There is strong evidence that GCs are a part of biological path integration system. In this thesis, we present a brand new accurate GC model based on a path integration mechanism, a cyclically connected artificial neuronal network, and a continuous attractor neuron network architecture. Our results show that GC model has high accuracy using continuous attractor models and a cyclically connected artificial neuronal network, which can be represented by a twisted torus, and allows the generation of regular triangular grids sheet across the environment based on inputs, which is only according to agent's velocity and heading direction. These tessellations share same spacing and orientation, as neighboring GC in the medial entorhinal cortex. A simple gain and bias mechanism allows to control the spacing and the orientation of grids sheet, which suggests that these different characteristics can be generated by a unique algorithm in the brain as inputs of place cells. A place cell is a kind of pyramidal neuron within the hippocampus that becomes active when an animal enters a particular place in its environment, which is known as the place field. Place cells are thought, collectively, to act as a cognitive representation of a specific location in space, known as a cognitive map. The difference between place cell and the grid cell is that, a grid cell will periodically fire in different places in the environment. Anatomical connectivity and recent neurophysiological results imply that GCs are the principal cortical inputs to place cells in the hippocampus. In this article a model is proposed, in which place fields of hippocampal pyramidal cells are formed by linear summation of appropriately weighted inputs from entorhinal GC and the connection between them is determined by the method of Hebbian learning.

Kurzfassung

Gitterzellen im entorhinalen Kortex der Ratte zeigen auffallend regelmäßige Feuerreaktionen auf die Position des Tieres im 2-D-Raum und es wurde angenommen, dass sie das neuronale Substrat für die Dead-Reckoning bilden. Es gibt starke Hinweise darauf, dass diese Zellen Teil des Integrationssystems für biologische Pfade sind. In diesem Artikel stellen wir ein zyklisch verbundenes künstliches neuronales Netzwerk vor, das auf einem Pfadintegrationsmechanismus und kontinuierlichen Attraktormodellen basiert und Gitterzellen auf einem simulierten mobilen Agenten implementiert. Unsere Ergebnisse zeigen, dass die synaptische Konnektivität eines Netzwerks basierend auf kontinuierlichen Attraktormodellen, die durch einen verdrehten Torus dargestellt werden kann, die Erzeugung regelmäßiger dreieckiger Gitter in der Umgebung basierend auf die Eingaben ermöglicht. Die Eingaben codieren nur die Geschwindigkeit und die Richtung des Agenten. Diese Tessellationen haben den gleichen Abstand und die gleiche Ausrichtung wie benachbarte Gitterzellen im medialen entorhinalen Kortex. Ein einfacher Verstärkungs- und Vorspannungsmechanismus ermöglicht die Steuerung des Abstands und der Ausrichtung von Gittern, was darauf deutet, dass diese unterschiedlichen Eigenschaften durch einen einzigartigen Algorithmus im Gehirn als Eingabe von Ortszellen erzeugt werden können. Eine Ortszelle ist eine Art pyramidenförmiges Neuron im Hippocampus, das aktiv wird, wenn ein Tier einen bestimmten Ort in seiner Umgebung betritt, der als Ortsfeld bezeichnet wird. Es wird angenommen, dass Ortszellen gemeinsam als kognitive Repräsentation eines bestimmten Ortes im Raum fungieren, der als kognitive Karte bezeichnet wird. Der Unterschied zwischen der Ortszelle und der Gitterzelle besteht darin, dass eine Ortszelle nur in einem bestimmten Bereich ausgelöst in einer Szene wird. Anatomische Konnektivität und neuere neurophysiologische Ergebnisse deuten darauf hin, dass Gitterzellen im medialen entorhinalen Kortex die wichtigsten kortikalen Eingaben sind. In diesem Artikel wird ein Modell vorgeschlagen, bei dem Ortsfelder von Hippocampus-Pyramidenzellen durch lineare Summierung entsprechend gewichteter Eingaben von entorhinalen Gitterzellen gebildet werden und die Verbindung zwischen ihnen durch die Methode des 'Hebbian Learning' bestimmt wird.

Abbreviations

HDCs	Head Direction Cells
GCs	Grid Cells
PCs	Place Cells
EC	Entorhinal Cortex
HC	Hippocampus
MEC	Medial Entorhinal Cortex
CANN	Continuous Attractor Neuron Network
AHV	Angular Head Velocity
LEA	Lateral Entorhinal Area
PER	Perirhinal Cortex
POR	Postrhinal Cortex
LTP	Longterm Potentiation
LTD	Longterm Depression
STDP	Spike-Timing-Dependent Plasticity

Contents

1. Introduction

Fast and accurate navigation to a predetermined target location (such as homing and foraging) is an important ability for animals to survive. And this ability is innate to most animals. Some previous biological experiments have shown that animals that have damaged hippocampus, enterorhinal cortex, and postsubiculum have lost the ability to navigate [1]. The ability to determine and update one's own position based on one's own motion information in a space environment is an important ability in biological navigation. It is in the medial entorhinal cortex (MEC) that the researchers discovered cells related to self-localization, such as the place cells (PCs) [2], grid cells (GCs) [3], head direction cells (HDCs) [4], border cells(BCs) [5] and speed cells (SCs) [6]. These cells in the MEC may determine how we perceive and remember where we are in the environment and the events we experience in that environment. GCs and PCs respond to the animal's location in the environment, BCs express the animal's proximity to geometric borders, SCs reflect the running speed of the animal, HDCs indicate the orientation of the animal relative to the angle speed and landmarks in the environment.

As the foundation of a biological navigation system, GCs are found to Periodically fire in relation to the animal's position in the environment and the firing map present a regular hexagon structure corresponding to the environment position [3]. It is the activities of these cells that animals can perceive their own position in the environment. After we gradually study the specific responses of these cell activities to the environment, it is important to utilize those biological-inspired PC models [7] and GC models [8] to realize the activities of these cells and fits the working methods of neurons. Among them, the realization of the GC model is particularly important. One of the most used neuron model of GC is the continuous attractor neuron network (CANN) [9]. Compared with other neuron models such as Leaky Integrate-and-Fire Neuron [10], this model focuses more on the activities of cell populations rather than on the activities of individual cells. This makes the model have better stability and anti-noise performance. And for CANN, we have the corresponding hardware to calculate [11]. This makes the entire neural model run efficiently, and the real-time performance of the calculation can be fully guaranteed.

Although there are many CANN-based GC models [8] [12], which use velocity of ainimals as the input to drive the GC model, they each have some their own problems, such as unreasonable structure, large size, low accuracy, difficult to run on corresponding hardware [11] and so on. For example, the driving method of GC model in [12] is unreasonable. Because in this model, the input speed information will directly and accurately changes the synaptic weights between neurons to achieve the purpose of driving the entire model. This method does not conform to the existing neural network models, such as neuron model in [9] [10], and it also violates the known real working mechanism of neurons in the brain. At the same time, because of this method this GC model cannot run on CANN-based hardware [11].

Therefore, there will be some obstacles in the future application of this GC model. In another model [8], although its structure and driving method are very reasonable. However, due to the lack of reasonable connections between the cells at the edges, it leads to great burden (128 $\times$128) for calculation, resulting in the lack of efficiency of the entire model. There are a very large number of different GCs in the navigation system, so that the animals can achieve the purpose of positioning in the environment. If only a GC model requires a lot of neurons to implement, then in future applications, this will be a problem that requires a lot of resource to drive the GC model. At the same time, under such a huge structure, the accuracy of this model is not good. As for accuracy, we can record the firing rate of GC to achieve the purpose of path integration. The difference between the estimated trajectory and the real trajectory indicates the accuracy of GC model. Under this GC model, the agent traveled 260 meters in 1800 seconds, resulting in an error of 15cm. This is not a very good result.

In this thesis, we are committed to proposing a more reasonable, more efficient and more accurate model. The high-performance GC model can be used in many fields with the support of corresponding hardware [11]. First of all, GC is an indispensable part of mammalian navigation system. Creating accurate and efficient models of it is significance to the research on biological navigation mechanisms. Second, The firing rate of GC is the input of many other models such as PC model. Only when there are enough GC models and these GC models are accurate enough, the PC model can be implemented. In the thesis, we also implement the PC model using traditional methods [13]. However, limited to the existing computing power of the computer, we used the ideal GCs outputs as the input of the PC model. This PC model has very good performance, which proves that the PC model can run well with a large amount of accurate GC input. In addition, the neuron network currently used in automatic driving and navigation requires a large amount of calculation, which results in high hardware requirements for the algorithm and a large energy consumption. At the same time, the real-time performance of the algorithm cannot be guaranteed due to the huge amount of calculation, however the real-time performance is very important for automatic driving and navigation. Some neuromorphic hardware [14] [11] is more in line with the real working mode of the biological brain. At the same time, it can solve the above problems well [15]. If we implement a CANN on hardware [11], it can complete very complex real-time calculations with less energy consumption. Therefore, the realizsation of efficient and precise GC and PC models based on CANN network will play a very important role in applying the CANN to automatic driving and navigation in the future.

Our main contributions of this thesis is that we have proposed a more reasonable, smaller, more efficient, and more accurate GC model.

- Our new GC model is completely based on CANN and combined with a 3-dimensional twisted ring structure. Different from the previous model [12], we newly added the shift layers structure, and the speed information as input directly stimulates the shift layer, thereby driving the entire GC model. And there is no need to change any structure of model, such as synaptic weight. Because our GC model strictly conforms to the CANN, it also can run on the corresponding hardware [11].

- Compared with another model [8], our model has smaller size yet higher accuracy. Only

$5 \times 18 \times 20$ cells are needed in our GC model. In our GC model, agent traveled 2246.44 meters in 8000 seconds, and only produced an about $3.41cm$ error in the end. So we only need to take up less hardware resources to achieve better results. This allows our GC model to have better performance in future applications. Therefore our GC model is a more reasonable and better model in future applications.

- In addition to our GC has a very good performance in the simulation environment, we applied this GC to a real robot. We use the data collected by the sensors on the robot as the input of the GC model, and the GC model also shows very good performance on the robot. This laid the foundation for our future application of GC to the field of robotics.

2. Biological Background

2.1. Spatial Cognition in Brain

Spatial cognition relates to cognitive processes in the brain required to support spatial localization and navigation in animals and humans. It is believed that the brain builds an internal cognitive map made up primarily of PC,GC and HDC as shown in Figure 2.1. Tolman [16] proposed the idea of rats having an internal cognitive map. These studies were followed by O'Keefe [17][18][2][19] who identified PC in the HC as the location of such a cognitive map. Later studies put forward the concept of integrating the external information in the environment and internal information into the rat as a part of the path integration system [20] [21][22][23][24][25]. More recently, Moser and Moser [26][3][27] identified the existence of GC located in the EC as part of a "neural odometry" system for rat navigation. Ranck [28] discovered the existence of HDCs in the brain. Together, place, grid and HDC are believed to provide for the underlying neural mechanisms required by the brain to build the cognitive map.

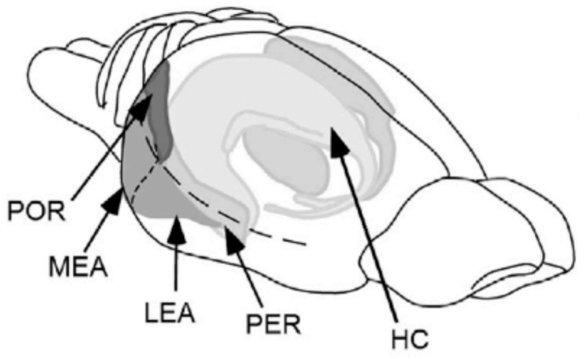

Figure 2.1.: HC,Entorhinal Cortices (LEA – Lateral Entorhinal Area, and MEA – Medial Entorhinal Area), Perirhinal Cortex (PER), and Postrhinal Cortex (POR) [25]

2.2. Place Cell

The experimental study of spatial representations in the brain began with the discovery of PC. More than 35 years ago, O'Keefe Dostrovsky [2] reported spatial receptive fields in complex-spiking neurons in the rat's HC, which are likely to be pyramidal cells [29]. Whenever the rat is in a certain location in the local environment (the place field of the cell), the cells in these locations will fire like in Figure 2.2. Neighboring cells fire at different locations, so in the entire HC, the entire environment was represented in the activity of the local cell population.That is to say, in a scene, which cell of PC fire represents a position of the agent in the environment corresponding to this cell. This position is an allocentric position in the environment [30][31]. The same participated in the representation for different environments, but the relationship of the firing field is different between one setting and another [17]. Inspired by Tolman [16], who suggested that local navigation is guided by internal "cognitive maps" that flexibly represent the overall spatial relationships between landmarks in the environment, and O'Keefe Nadel [17] proposed that PC is the basic elements of a distributed allocentric map representation. PC is suggested to provide the animal with a dynamic, continuously updated representation of allocentric space and the animal's own position in that space. We now have abundant evidence from a number of mammalian species demonstrating that the HC plays a key role in spatial representation and spatial memory [32][33][34].

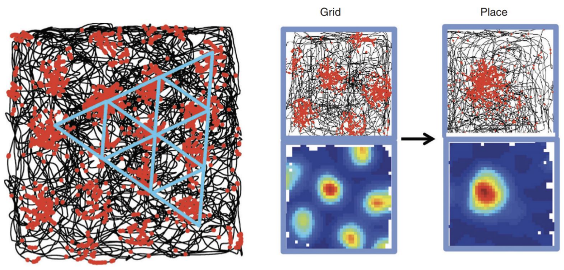

Figure 2.2.: GC and PC. (Left) A GC from the EC of the rat brain. The black trace shows the trajectory of a foraging rat in part of a 1.5-m-diameter-wide square enclosure. Spike locations of the GC are superimposed in red on the trajectory. Each red dot corresponds to one spike. Blue equilateral triangles have been drawn on top of the spike distribution to illustrate the regular hexagonal structure of the grid pattern. (Right) GC and PC. (Top) Trajectory with spike locations, as in the left part. (Bottom) Colorcoded rate map with red showing high activity and blue showing low activity. GC are thought to provide much, but not all, of the entorhinal spatial input to PC [35].

2.3. Head Direction Cells

The activity of HDC found in the brains of freely moving rats have remarkable properties. Regardless of the animal's position in the environment, they will signal the instantaneous head direction of the animal on the horizontal plane. [28][36] in Figure 2.3. The amazing thing about this system is that its frame of reference is completely centered on the world, so it can actually be used as a neural compass or gyroscope. In other words, we can obtain the specific world direction of the agent in the environment through HDC activities. Even in a completely dark environment, the internal representation of the head orientation maintained by these cells is constantly updated according to the animal's head movement [37][38]. The cell's functionality can be maintained without visual cues, only using self-movement information. This process is called path integration. Angular Head Velocity (AHV) cells which respond to the animal's angular velocity were also found and could play a role in driving the HDC's behavior [39].Usually they are used as input of HDC. In the updated research, we found that HDC can use familiar landmarks to reset or calibrate the internal direction [36] [40] [41] [42]. But there are some restrictions. McNaughton [43] hypothesized that visual cues exert control over the direction sense only after the rat has learned a stable mapping between the visual information and the head direction information.

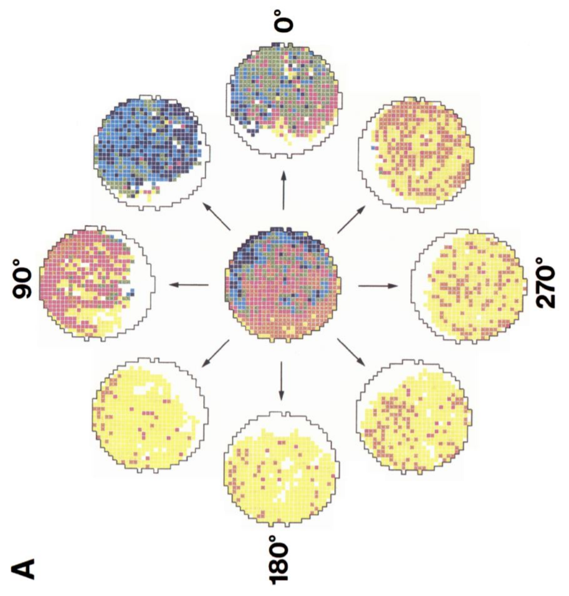

Figure 2.3.: Color-coded direction-specific firing rate maps for one head-direction cells. The maps set at the vertices of an octagon represent the spatial firing pattern that was observed while the animal's head pointed in the 45° range centered on the arrows pointing toward each map. The map is for 0°, it is centered at the 3 o'clock position, with increasing degrees (and map numbers) proceeding counterclockwise around the octagon. The map in the middle is a composite firing rate map (i.e., direction-independent) that shows the overall average spatial firing pattern. The regions of low firing in the composite maps are the result of the animal's inability to put its head into that region while the head points in the preferred direction. Median firing rates for color-coded rate categories are given in the order orange, red, green, blue, purple. The firing rate for yellow-coded pixels was zero. All firing rates are in spikes/sec. A, Direction-specific maps: 1.9, 15.0, 51.4, 85.0, 111.0. Composite map: 3.5, 10.0, 16.9, 25.7, 42.0. [36]

2.4. Grid Cells

All subfields of the hippocampal region contain place-modulated neurons, but the most distinct firing fields are found in the CA areas [44], which is shown in Figure 2.1, on the basis of the apparent amplification of spatial signals from the EC to the CA fields [45]. Until recently, many researchers believed that place signals mainly depended on calculations in the HC network. This view was challenged by the observation that spatial firing persisted in CA1 neurons after removal of intrahippocampal inputs from the dentate gyrus [46] and CA3 [47]. This raised the possibility that spatial signals were conveyed to CA1 by the direct perforant-path projections from layer III of the EC. Projection neurons in layers II and III of the medial entorhinal cortex (MEC) were subsequently shown to exhibit sharply tuned spatial firing, much like PC in the HC, except that each cell had multiple firing fields [26]. The many fields of each neuron formed a periodic triangular array, or grid, that tiled the entire environment explored by the animal [3] in Figure 2.2. Such GC collectively signal the rat's changing position with a precision similar to that of PC in the HC [26]. The graphics paper–like shape of the grid immediately indicated GC as possible elements of a metric system for spatial navigation [3], with properties similar to that of the allocentric map proposed for the HC more than 25 years earlier [17].

Each grid is characterized by spacing (distance between fields), orientation (tilt relative to an external reference axis), and phase (displacement relative to an external reference point). Although cells in the same part of the MEC have similar grid spacing and grid orientation, the phase of the grid is nontopographic, i.e., the firing vertices of colocalized GC appear to be shifted randomly, just like the fields of neighboring PC in the HC. The spacing increases monotonically from dorsomedial to ventrolateral locations in MEC [3], mirroring the increase in size of place fields along the dorsoventral axis of the HC. Cells in different parts of the MEC may also have different grid orientations [3], but the underlying topography, if there is one, has not been established. Thus, we do not know whether the EC map has discrete subdivisions. The EC has several architectonic features suggestive of a modular arrangement, such as periodic bundling of pyramidal cell dendrites and axons and cyclic variations in the density of immunocytochemical markers [48], but whether the anatomical cell clusters correspond to functionally segregated grid maps, each with their own spacing and orientation, remains to be determined.

3. Related Work

Below I will introduce the main implementation methods of PC, HDC and GC. And a neural model on which these methods are based.

3.1. Continuous Attractor Neural Network

Continuous attractor neural network (CANN) is an important method for us to realize GC, HDC and PC. 'Attractor' refers to the active stationary state that a dynamic system can maintain by its own dynamics without receiving external input. It is widely regarded as the way the nervous system expresses information. Of course, the real neuron system does not require a stable state in the strict sense, and the attractor network is just an idealized mathematical model used for research convenience. Interestingly, unlike the usual experimental findings promoting theoretical research, the development of CANN's theory is ahead of the experiment. As early as the 1970s, several pioneers of computational neuroscience Wilson, Cowan, and Amari proposed the CANN model from a purely theoretical point of view and studied its mathematical properties [49] [50]. In the 1980s, more mathematics and physicists joined in, which further promoted the study of the dynamic properties of CANN [51][52]. However, the wide application of CANN in neural information processing modeling began after the 1990s. Classic successful examples include the mental rotation phenomenon of motion direction coding [53], visual orientation tuning [54], and head orientation coding [55] etc. These modeling work proves the importance of CANN in neural information. The most direct experimental evidence for CANN has appeared in recent years [56][57][58] [59].

3.1.1. Mathematical Model of Continuous Attractor Neural Network

To form an attractor network, two basic conditions are required: First, there are excitatory recurrent connections between neurons. Without external input, it depends only on the positive feedback between neurons, the network can maintain stable activities. At the same time, we also require the excitatory connection to be local, so as to form meaningful spatial local activities. second, the network must have an inhibitory effect, so as to prevent the system activity from exploding due to repeated positive feedback. After adjusting the ratio of excitement and inhibition, an attractor network is formed. The Amari-Hopfield model uses the idea of attractors to explain the associative memory mechanism of the brain under the condition of receiving partial or fuzzy information [60]. However, the classic Hopfield model does not consider the symmetrical structure of the connections between neurons, so the attractors are isolated from each other in space. On the basis of the attractor network, if we further require that the connections between neurons have a symmetrical structure with spatial translation invariance, then the network will have a cluster of continuous attractors instead of isolated. Considering the real The number of nerves in a biological system is large enough. For convenience, the following discussion assumes that the number of neurons is infinite, these attractors are closely arranged in the parameter space to form a continuous state subspace. This is where the name "continuous attractor" comes from [52]. Mathematically, it is easy to construct a high-dimensional CANN, but due to the limitation of the two-dimensional cerebral cortex on the distribution of neurons, usually we only use one-dimensional or two-dimensional CANN. Birds due to the need of flight, their spatial positioning system may use 3D CANN. There are several neural models of CANN, which will be introduced in the following chapters.

3.1.2. The Dynamic Properties of Continuous Attractor Neural Network

The special structure of CANN determines its special dynamic properties, which in turn determines the calculation functions it can achieve. We can intuitively understand CANN's unique dynamic properties. We consider HDC. An HDC has a cluster of Gaussian wave packets in the steady state (Figure 3.1 A), which form a one-dimensional energy-smooth subspace in the system state space (Figure 3.1B). In this subspace, because the energy function is smooth, the system state is in neutral stability, which means that the system can easily change its state under the drive of a small external input[61] [62]. Note that this property is not possessed by other discontinuous attractor networks such as Hopfield networks. This neutral stability is the key to CANN dynamics. It enables the network state to smoothly track the external motion input, thus realizing multiple calculation functions.

So we sum up two characteristics of cann: First, it is stable when there is no additional output, and this is its stability. Second, his stable state is very easily changed by external input. This is its sensitivity. These two characteristics make it very suitable for the realization of PC, HDC and GC.

3.1.3. Calculation Function of Continuous Attractor Neural Network

The reasonable biological structure and dynamic properties of CANN make it a commonly used model to simulate the process of neural information processing. There is a lot of neural modeling work related to CANN. I will give some examples to illustrate.

3.1.3.1. Neuron Group Coding

Experiments have found that for many types of stimuli, especially continuous variables, such as angle, spatial position, etc., the coding strategy of the neuron system is: a large group of neurons work together to code the stimulus value, and the response of each neuron covers a certain range of stimulus value and it has the maximum response to a specific value (shown as a Gaussian tuning function) in Figure 3.1. the tuning function of the neuron group covers the entire parameter space. When an external stimulus is presented, a group of neurons react to average out the ubiquitous noise. CANN just provides a very natural network implementation mechanism for this group coding strategy: the neuron group involved in coding is connected to form a CANN. when there is external input, the network enters an attractor, that is, a Gaussian wave packet is generated. Because it is an attractor, the noise is removed, and the vertex position of the wave packet gives the result of neural network decoding [63][64]. Theoretical analysis shows that the decoding algorithm implemented by CANN is "template matching", which can also be regarded as "unfaithful decoding" while ignoring neuron-related activities [65][66]. This is also the main method we used in HDC, GC and PC.

3.1.3.2. Multimodal information integration

The brain uses a variety of sensory organs such as eyes, ears, nose, touch and balance to perceive the external world. These sensory organs extract information from external stimuli through different physical, chemical, mechanical and other signals. This information should be integrated in the brain to help us understand the external world more reliably and comprehensively. A large number of behavioral experiments have shown that the brain does integrate and optimize multi-modal information. From a computational point of view, when integrating these signals with different attributes or characteristics, the brain needs a common information expression mode to exchange information with each other. CANN, in which information is uniformly expressed as neural group activity wave peaks, provides a possibility for multi-modal information integration. Wu studied the neural network model that integrates vision and balance information are inputs of the heading direction model [67]. Based on the experimental data, a decentralized computing model is proposed, in which CANN on first CANN mainly accepts visual signals, while another CANN mainly accepts balanced signals, and the two CANN exchange information through a long-distance connection. Recent research shows that two coupled CANNs can achieve Bayesian optimization of information integration and statistical optimization of information separation[68][69]. CANN can be used for information expression, storage, calculation, and communication in the brain Unified network framework.

This provides the possibility for us to use vision, balance and other information to Calibrate GC and HDC in the future.

3.1.3.3. Hardware Implementation of Continuous Attractor Network

Since CANN has a biologically reasonable structure and powerful computing functions, it is not difficult to imagine that scientists will try to implement it on hardware. Indeed, as early as 2000, MIT's Seung and his colleague had implemented CANN with simple circuits, but due to the limitations of the technology at that time, only a dozen neurons were used, and no truly valuable functions were realized[11]. In recent projects, brain-like computers already have tens of millions of neurons and synapse[70]. It is precisely because of the gradual power of brain-like computers that our calculations can be negligible fast. This is also the biggest advantage of the third-generation neural network (spike neural network).

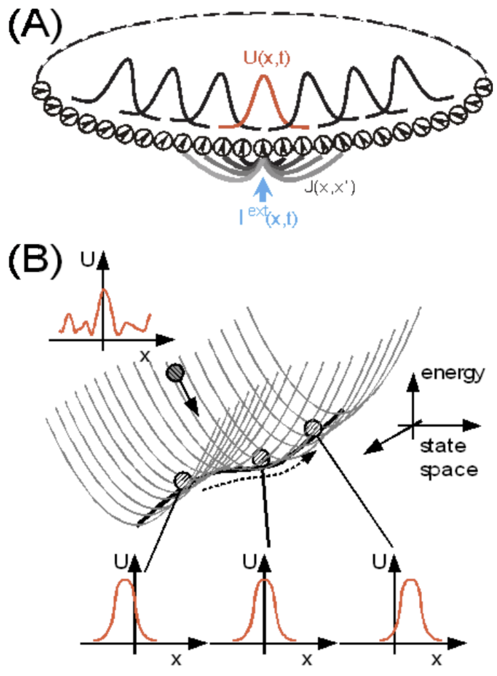

Figure 3.1.: **A continuous attractor neural network (CANN) model.**(**A**) An illustration of a one-dimensional CANN, which encodes a continuous variable (e.g. orientation or direction) x in the region of $(-\pi, \pi]$ with the periodic condition. Neurons are aligned in the network according to their preferred stimuli. The neuronal connection pattern $J(x, x')$ is translation-invariant in the space. The network can hold a continuous family of bump-shaped stationary states (**B**). The stationary states of the CANN form a subspace in which the network states are neutrally stable. The subspace is illustrated as a canyon in the state space of the network. The movement of the network state along the canyon corresponds to the position shift of a bump.

3.2. Spatial Cognition Modeling

Various spatial cognition models have been developed in the past 30 years. Most of these models are based exclusively on PC models, while some provide some level of integration with GC and HDC. These models are further discussed in [71] [72] [73][74][75][76][77][78][79][80]. These models evaluate different aspects of spatial cognition including in some cases using both simulation and robotics experimentation. The rest of this section describes a spatial cognition model as a example, which is summarized in Figure 3.2. In the latest research,

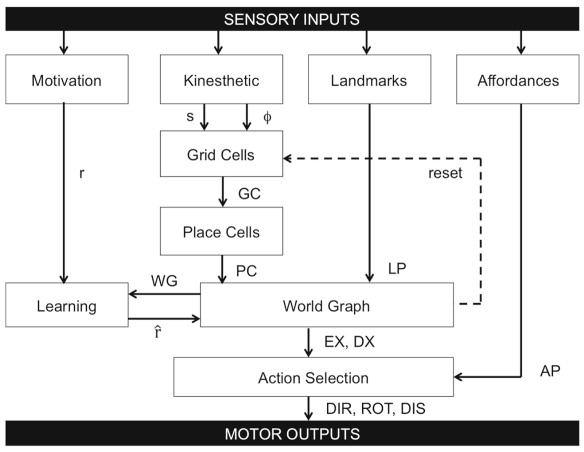

Figure 3.2.: The diagram describes an extension to the original Barrera and Weitzenfeld [72] spatial cognition model.

An important aspect of spatial cognition model is that it relates spatial cognition processes including learning and memory by:

- Interaction of different brain structures to demonstrate skills associated with global and local localization in space. Such as the activity of GC are inputs of PC.
- Path integration by GC.
- the use of kinesthetic and visual cues during orientation.
- generation of topological-metric spatial representation.
- adaptation using Hebbian learning[81], for example in PC.
- Management of rewards by use of reinforcement learning using an Actor-Critic architecture[82].The learning architecture is complemented by applying backward reinforcement to successful routes followed by the agent during training thus enabling learning of explored routes. After exploration, the model exploits maximum reward expectations to guide the agent towards the goal from any given departure location. Additionally, the model implements an on-line learning process to adapt the cognitive map to changes in the physical configuration of the environment.

Here are a few models that are frequently mentioned and applied in this article

- HDC module receives kinesthetic information in the form of agent's angular speed. Additionally, a reset mechanism provides recalibration feedback from the World Graph to the HDC module to prevent HDC activity from drifting over time. HDC provides GC with the direction of the agent in the environment.
- GC module, corresponding to the EC. It receives kinesthetic information in the form of agent speed and orientation. Additionally, a reset mechanism provides recalibration feedback from the World Graph to the GC module to prevent GC activity from drifting over time. Compared with the existing industrial navigation system, GC is equivalent to being used as an odometer. And it can use visual and other information for calibration.
- PC module, corresponding to the HC. it receives input from GC. Connections between different PC layers are strengthened by Hebbian learning[81]. It also represents the world coordinate position of the agent in the local environment.

3.3. Head Direction Cells Modeling

Among the HDC models I know, the one-dimensional CANN model is used more. In this model, each cell represents a preferred direction. HDC are cells sensitive to a preferred head direction, they spike maximally at their preferred orientation. The CANN model was first described in [83] to explain directional tuning in HDC. Although there are many models of HDC, many of them have similar structures in Figure 3.3. Among them, we can see that the activity of HDC presents a peak. The preferred direction corresponding to this peak is the direction of the agent in the environment. The connection strength between HDC neurons is very clearly explained in [55]. It is also the basis of many HDC models.

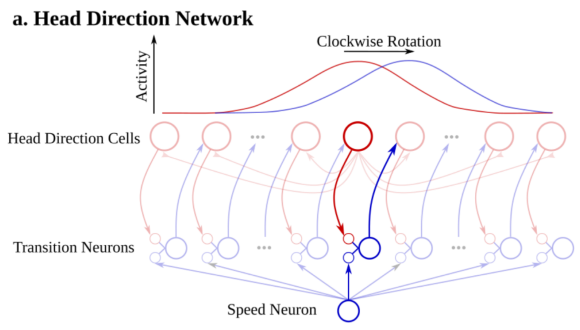

Figure 3.3.: Head direction network with one group of transition neurons[84]

3.4. Grid Cells

GC are located in the MEC and have arbitrary field sizes and spacing. Mapping GC activity onto the respective trajectory creates a hexagonal pattern. An essential characteristic of this hexagonal pattern is the independence of its speed and head direction, ie. the distance between fields remains constant even across environments. Given input from speed cells and HDC, GC encode the relative position, ie. the distance and direction between two reference positions. Both speed cells and HDC are among other locations situated in the MEC as well and are functionally constant across the environment. Speed cells adjust the frequency of firing based on the speed of movement. The input of the GC model is the linear velocity of the agent and the direction of the HDC Figure 3.4. In this way, we can get the linear velocity of the agent in the world coordinate system. Below I will introduce several common models for implementing GC.

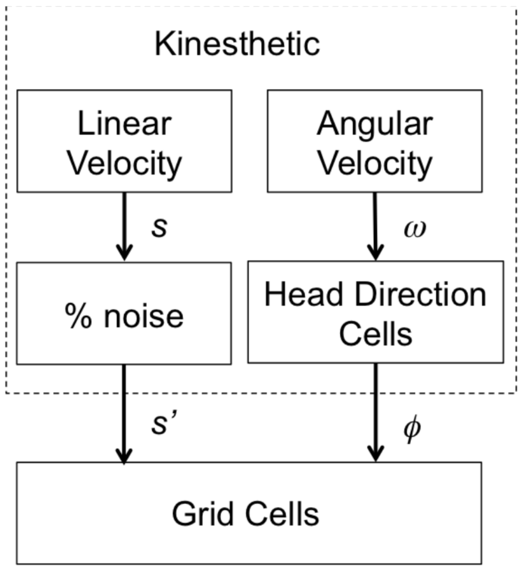

Figure 3.4.: The input of GC

- **Attractor Network Models** The earliest and the basis of many HDC is from these two articles[86][55]. In these models, the cells are arranged in a ring structure. Each cell has a corresponding direction. Its neighbors generally give it a positive connection, and the strength of this connection gradually decreases with the increase of the distance between them, and eventually becomes a negative strength connection. At this time, the shape of the HDC activity is shown in Figure 3.3. When we design an additional input to HDC to offset the connection between HDC, then the entire peak will also move, and so the network is able to effectively track the orientation of the head. With the subsequent discovery of GC, the 1-D ring attractors were extended to two dimensions[21]. The CANN structure of GC will be explained in detail in later chapters.

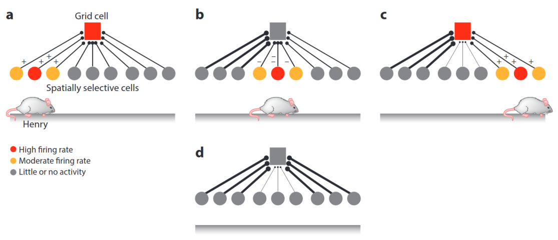

Figure 3.5.: Adaptation model for GC. The figure shows changes in the synaptic weights as an animal navigates a linear track according to the adaptation model. In each panel, the circles represent spatially selective cells (e.g., PC in the HC), and the square represents a GC. Red indicates high firing rates, yellow indicates moderate firing rates, and gray indicates little or no activity. (a)As the animal starts on the left side of the track, the second cell from left, which is most selective to the current position of the animal, is maximally active, whereas its two neighboring cells are less active, and those more to the right of the track are inactive. If the initial connections are all the same, this leads to firing of the GC and strengthening of the couplings between the active spatially selective cell and the GC, as indicated by the plus signs next to the connections. (b) As the animal moves to the right of the track, a different set of cells become active, but their activity does not lead to GC activity, as the GC is now adapted after being active in panel a.This consequently leads to suppression of the synaptic weights from the active spatially selective cells to the GC, as indicated by minus signs near the connections. (c) The GC has now recovered from adaptation, and the same scenario as in panel a is now repeated for the cells to the rightmost part of the track. (d) Repeated traversals along the track lead to a mature system, with strong and weak connections represented by the thicknesses of the lines. A similar mechanism of adaptation will make neurons selective for regularly distanced positions in two dimensions: Periodic bouts of activation and fatigue cause some spatial cell–to–GC connections to strengthen and others to weaken, which in turn creates circular fields with minimal interfield spacing (i.e., a hexagon). In the real world, the alternating periods of activation and fatigue will not occur in precisely the same place every time because of behavioral variability, but this is averaged out over multiple runs through the environment. Consistently faster running in a given direction when the grid network is formed in developing animals might lead to architectures that support elongated grid[85]

- **Adaptation Models** In its simplest form, this type of model relies on the idea that a single cell can generate a grid pattern. In the simplest version of the model[5], a single GC receives convergent input from PC inputs that follow simple Hebbian learning rulesFigure 3.5. The key part of the model is GC adapt to spike frequency, creating the required competition suggested by Turing. As a result of this simple arrangement, the model MEC cells gradually acquired a hexagonal emission pattern, The spacing of the grid pattern depends on the peak frequency adaptation. During training an adaptation mechanism ensures that the connections from sets of place fields are alternately strengthened and weakened. The GC first adapts to a set of place fields While traversing the next set of place fields the adaptation to the previous set inhibits the adaptation mechanism while gradually decreasing itself . But I personally think this method is far-fetched. Because if there is no GC input, how can there be PC activity.
- **Oscillatory-Interference Model** According to [87] basic single cell model consists of two or more oscillators: a baseline oscillator oscillating at a constant frequency and others varying its frequency of oscillation based on speed. Then compare the interference modes of the two oscillators with the threshold, and the unit will fire if the threshold is exceeded. This in turn means that when some oscillators are in-phase the cell spikes. this model can be extended by using dendritic oscillators at 60 degree angles to create the hexagonal firing pattern. A major problem with this more complex model is phase lock, which can be solved by using neuron populations instead of compartments[88]. Another shortcoming of this model is the lack of interdependence between GC[88].

3.5. Place Cells

according to [7] We can know that the input of PC is GC. We can determine the agent's position in the entire environment by GC with different spacing, orientation and phase in [89] in Figure 3.6. The connection strength between GC and PC, PC and PC can be obtained by Hebbian learning.

3.5.1. Hebbian Learning

Hebbian theory is a neuroscience theory that explains the changes in neurons in the brain during the learning process. This theory [90] describes the basic principle of synaptic plasticity, that is, the continuous and repeated stimulation of presynaptic neurons to postsynaptic neurons can lead to an increase in the efficiency of synaptic transmission. That is to say, the continuation and repetition of the fire of neurons will lead to the lasting improvement of neuron stability. When the axon of neuron A is very close to neuron B and participates in the repeated and continuous excitement of B, these two neurons or one of them will undergo certain growth processes or metabolic changes, causing A to be one of the cells that can excite B, and at the same time the connection between them becomes more strong.

This theory also serves as the basis of the Spike Neural Networks and is the basic learning theory of Spike Neural Networks. According to the different neural models, we mainly have

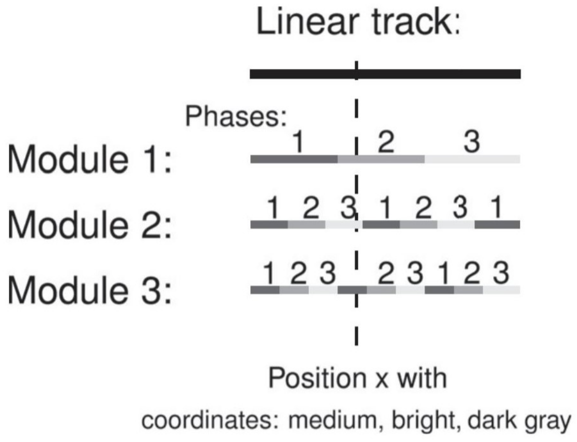

Figure 3.6.: Nested interval scheme. Example with three clearly different spatial periods and three discrete phases each. The first module gives coarse spatial information that is further refined by the other two modules. By themselves, the other modules provide ambiguous spatial information on the range; together, they effectively subdivide the unit interval.

two Hebbian learning rules. The first is that the entire neuron system is completely based on impulse signals and use integrate-and-fire neurons[91]. The other is a neuron system based on the firing rate of cells[92].This model pays more attention to the activities of neuronal population.

3.5.1.1. Spike-Timing-Dependent Plasticity

Longterm Potentiation (LTP) describes the longterm strengthening of a synapse based on co-activity of pre- and postsynaptic neurons within a certain time window and with the contribution dependent on the temporal proximity of co-activity. While Longterm Depression (LTD) describes the longterm weakening of synaptic strength [10]. Spike-Timing-Dependent Plasticity (STDP) makes use of LTP and LTD dependent on the relative spike times of pre- and postsynaptic neurons.

According to [93], in STDP strenghtening and weakening is modelled by a scalar weight parameter in Figure 3.7.

$$\Delta w(\Delta t) = \begin{cases} A_{+} \exp\left[\frac{\Delta t}{\tau_{+}}\right] & \Delta t < 0 \\ -A_{-} \exp\left[\frac{-\Delta t}{\tau_{+}}\right] & \Delta t > 0 \end{cases} \tag{3.1}$$

Where $\Delta t = t_{pre} - t_{post}$ is the interval between pre- and postsynaptic spike times, with t_{pre} is the last spike time of presynaptic and t_{post} is the last spike time of postsynaptic. A_{+}, A_{-} are the scaling factors and τ_{+}, τ_{-} are the time constants for LTP and LTD window. That is to say when $\Delta t < 0$, $\Delta w > 0$ the synapse is strengthened. On the contrary, when $\Delta t > 0$, $\Delta w < 0$ the synapse is weakened. In summary, when the last spike time of presynaptic is before the last spike time of postsynaptic, the synapse is strengthened, Means that the spike of presynaptic cause the spike of postsynaptic.

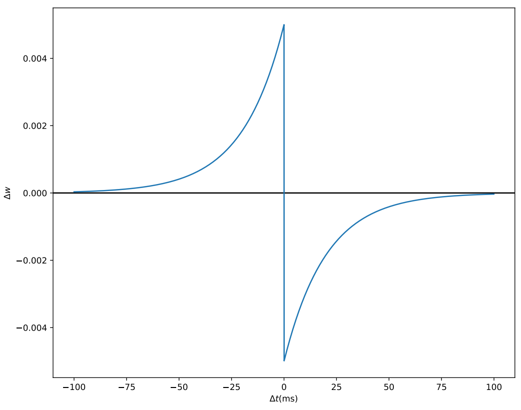

Figure 3.7.: Weight modifications based on STDP. $A_+ = A_{=0.005}$, $\tau_+ = \tau_{=20ms}$. $\Delta t < 0$ shows weight modifications based on LTP. $\Delta t >= 0$ shows weight modifications based on LTD.

3.5.1.2. Hebbian Synaptic Rule Based on Firing Rate Model

Another network that uses firing rate as the basis of neuron model in section 4.1 will have another learning rule. There are a lot of learning rules based on the firing rate, I just introduce two of them in this section. These rules do not emphasize the determination of causality between cells through learning. As mentioned above,like A can make B fire. Because they don't consider spike time at all, only the firing rate of the cell. When their firing rates are both large at the same time, we will strengthen the strength of the connection between them. If the opposite is the case, it will reduce the strength of the connection between them.

- **Spike product rule**: A learning rule according to [13]:

$$w_{ij}(t+1) = w_{ij}(t) + c\overline{x}_i(t)\overline{x}_j(t) \tag{3.2}$$

  where $\overline{x}_i(t)$ and $\overline{x}_j(t)$ are the mean firing rates of the post- and presynaptic neurons like in section 4.1 and c is a proportionality constant that determines the learning rate. The three key assumptions in Equation 3.2 are that the contributions of pre-posttsynaptic activity can be separated into two corresponding activity terms, that the proper measurement of activity is firing rate, and that the nature of the interaction between pre- and postsynaptic firing rate is a simple product.

- **Presynaptically Gated Rule:** A learning rule according to [89]:

$$\frac{dw_{ij}^{gp}}{dt} = k(\overline{x_j} - \theta_p)\overline{x_i} \tag{3.3}$$

Where w_{ij}^{gp} is the weight, $\overline{x}_i(t)$ and $\overline{x}_j(t)$ are the mean firing rates of the post- and presynaptic neurons, k is the learning rate factor and θ_p is a threshold on the presynaptic firing rate. Equation 4.29 is theoretically more reasonable than the previous rule Equation 3.2. Because in this rule, the weight is easier to reach a stable state and postsynaptic activity is required to trigger synaptic modification. The direction of change is determined by the presynaptic firing rate: if presynaptic firing rate is greater than θ_p, the synapse is potentiated; otherwise, the synapse is depressed.

4. Methodology

In this chapter, we will introduce our specific GC and PC implementation methods. Including their neuron model, the structure of the model and the operating method of these models.

4.1. Neuron Model

Biological neuron models, also known as spiking neuron models [10], are mathematical descriptions of the properties of certain cells in the nervous system that generate sharp electrical potentials across their cell membrane, roughly one millisecond in duration, called action potentials or spikes. Since spikes are transmitted along the axon and synapses from the sending neuron to many other neurons, spiking neurons are considered to be a major information processing unit of the nervous system. Spiking neuron models can be divided into different categories: the most detailed mathematical models are biophysical neuron models (also called Hodgkin-Huxley models [94] [10] [95]) that describe the membrane voltage as a function of the input current and the activation of ion channels. Mathematically simpler are integrate-and-fire models [96][97] that describe the membrane voltage as a function of the input current and predict the spike times without a description of the biophysical processes that shape the time course of an action potential. Even more abstract models only predict output spikes (but not membrane voltage) as a function of the stimulation where the stimulation can occur through sensory input or pharmacologically. At the same time, Some neuron models as mentioned in subsubsection 3.1.3.1, the CANN section 3.1 focuses on neuron group coding and is based on rate-based neuron model, and the rate-based neuron model has better performance against noise [64]. GC and PC need better accuracy and noise immunity, therefore we do not use spike-based neuron model [94][10][95] but rate-based neuron model [63][61][8][12]. In this thesis we use the rate-based neuron model of the attractor network in [12]. In a real biological system, we generally count the number of cell fires within a period to obtain the firing rate of the cell. The exact spike timing was not shown to be relevant for the information represented by GC and is ignored [63][61][8][12]. The rate-based neuron model in [12] we used is defined as: The activity of a cell i at time $t+1$,i.e. $A_i(t+1)$ is defined using a linear transfer function $B_i(t+1)$ given by:

$$B_i(t+1) = \sum_{j=1}^{N} A_j(t) w_{ij} + I_i \tag{4.1}$$

, where $A_i(t)$ is the firing rate of neuron i at time t, $B_i(t+1)$ is the net inputs or "synaptic currents" of neuron i at time $t+1$, N is the number of cells connected with the neuron i, w_{ij} is the synaptic weight connecting neuron j to cell i, with $i, j \in \{1, 2, \ldots, N\}$. I_i is a external input to receive special inputs,which will mention in section 4.4, or stimulus. A normalization mechanism from the synaptic currents to neuron's firing rate can ensure the stability of the network. Thus:

$$A_i(t+1) = B_i(t+1) + \tau \left(\frac{B_i(t+1)}{\sqrt{\sum_{j=1}^{N} B_j(t)}} - B_i(t+1) \right) \tag{4.2}$$

, where constant parameter τ determines the stabilization strength. To implement this mechanism locally, we use an additional cell (cell $N+1$which is not a GC), that computes the sum of all synaptic currents of the neurons. Normalized by that, the sum of synaptic currents of these external neurons $\sqrt{\sum_{j=1}^{N} B_j(t)}$ is transferred back to the neurons of the network. In order to prevent negative, we set:

$$A_i(t+1) = \begin{cases} A_i(t+1) & A_i(t+1) > 0 \\ 0 & \text{others} \end{cases} \tag{4.3}$$

In our entire model, all cells follow this neuron model.

4.2. Network Architecture

We will introduce the network's basic structure in this section. Our entire network is divided into five layers: value space layer,shift left layer, shift right layer, shift up layer and shift down layer,which is shown in Figure 4.2. For the value space layer, its activities represent the activities of the entire GC. We can decode the information we need from the activities of cells in this layer. The firing rate of cells will be formed as a peak, which is shown in Figure 4.7. The other four layers are the shift layers, which are used to move the peak in the value space layer. We will introduce the specific structure of these five layers of neurons in detail, their internal connection method and the connection method between layers in following sections.

4.2.1. Value Space Layer

Value space layer is a very important layer of neurons in this GC model. It is the structure of this layer that enables us to achieve static GC and hexagonal firing pattern. The formation of stable spaced firing peak fields is attributed to Mexican-hat connectivity, a type of connectivity in which cells in value space layer are assumed to receive excitatory inputs from cells with similar spatial phases, creating a local excitatory hill, and inhibition at larger phase differences. The competition between short-range excitation and longer-range inhibition imposes a stable peak of activity of cells in the layer. In this thesis, we use the Gaussian weight function as the synaptic weight function: Figure 4.2:

$$w_{ij} = I \exp\left(-\frac{d_{ij}^2}{\sigma^2} \right) - T \tag{4.4}$$

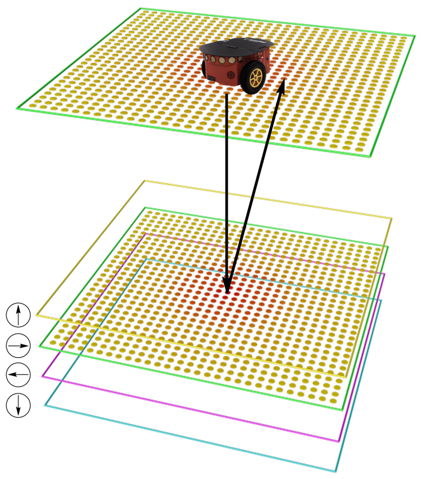

Figure 4.1.: GC model structure with shift layers: The top layer is value space layer. The under four layers respectively are shift up, shift right, shift left and shift down layers.

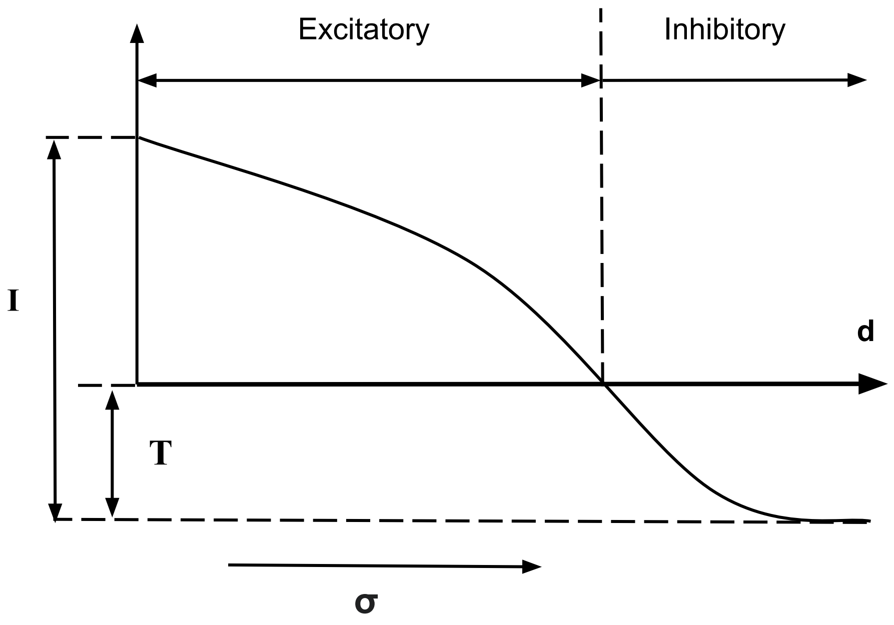

Figure 4.2.: Synaptic weights connecting cell to other cell wich geometric distance d. I is the intensity parameter, σ regulates the size of the Gaussian function, T is the shift parameter determining excitatory and inhibitory connections.

Where d_{ij} is the geometric distance between cell i and j, w_{ij} is the synaptic weight between cell i and j. $I$is the intensity parameter, defining the overall synaptic strength, σ regulates the size of the Gaussian function, T is the shift parameter determining excitatory and inhibitory connections.

In order to produce a periodic hexagonal structure, we need to give a detailed definition of the geometric distance between cells. In this thesis we use a twisted torus structure from [12] in Figure 4.3. It can be imagined that our neurons are on the surface of this twisted torus

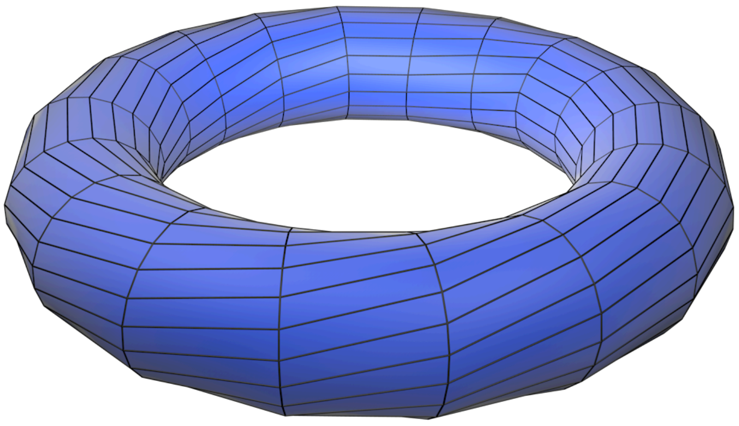

Figure 4.3.: 3-d twisted torus model of grid cell:The intersection of each horizontal line and vertical line represents a cell in a value space layer. The shortest distance between the two cells on the surface of this 3-d structure represents the distance between the two cells. It is this ring structure that ensures the periodic firing of the GC.

structure as shown in Figure 4.3 from [12]. And the shortest distance on the surface between cell i and cell j is the d_{ij}. Now we need to build this twisted torus model in a two-dimensional space. First, we use the horizontal as the x-axis and the vertical as the y-axis. In order to meet the hexagonal structure, we need the number of neurons in the y direction to be Probably $\frac{\sqrt{3}}{2}$ than the number of neurons in the x direction, which is shown in Figure 4.4. We constructed a population of $N = N_x \times N_y$ neurons, which organize in a grid sheet matrix covering the repetitive rectangular structure. At the same time, we need to comply with:

$$N_y \approx \frac{\sqrt{3}}{2} N_x \tag{4.5}$$

4.2.2. Synapses in Value Space Layer

Now we have determined the relationship between the synaptic weight and the distance between two cells in Equation 4.4, and also determined the neuron model we need to use in

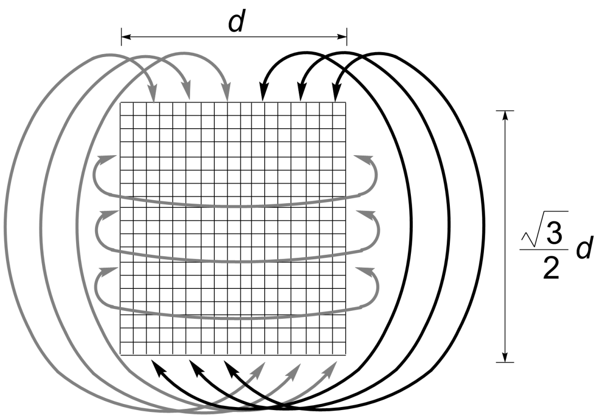

Figure 4.4.: Matrix of a population of 20 × 18 GC. The arrows represent the connection between the GC cells on the edge. For example, the cells on the lateral edge are directly connected to the cells on the other side.

section 4.1. Next, we need to define the distance between each cell on the 2d grid sheet, so that this distance meets the geometric distance between each cell in the 3-dimension structure Figure 4.3.

For this grid sheet, we still use the definition in [12]. The weight between two cells in the grid sheet like Figure 4.4 is defined by:

$$d_{ij}^2 = \| c_i - c_j \|_{tri}^2 \tag{4.6}$$

Where $c_i = (c_{ix}, c_{iy})$ is the position of the cell i, defined by:

$$\begin{aligned} c_{ix} &= (i_x - 0.5)/N_x \\ c_{iy} &= \frac{\sqrt{3}}{2}(i_y - 0.5)/N_y \end{aligned} \tag{4.7}$$

Where $i_x \in \{1, 2, \dots, N_x\}$, $i_y \in \{1, 2, \dots, N_y\}$. i_x and i_y are the column and the row number of cell i in 2-dimension matrix grid sheet. The norm $\| \cdot \|_{tri}$ defines the induced metric distance of the network. To obtain the 3-dimensional twisted ring structure of the value space layer in Figure 4.3, the border of the layer have to be the neighbors of the cell at the opposite border, which is shown in Figure 4.4. This two-dimensional grid sheet structure in Figure 4.4 corresponds to the three-dimensional structure in Figure 4.3. In fact, induced metric distance of the 2d matrix grid sheet network describes the shortest distance between each cell on the surface of the three-dimensional structure Figure 4.3. The following is the specific definition of $\| \cdot \|_{tri}$ from [12]:

$$\| c_i - c_j \|_{tri} = \min_{j=1}^{7} \| c_i - c_j + s_j \| \tag{4.8}$$

Where

$$\begin{aligned} \mathbf{s_1} &= (0,0) \\ \mathbf{s_2} &= \left(-0.5, \frac{\sqrt{3}}{2}\right) \\ \mathbf{s_3} &= \left(-0.5, -\frac{\sqrt{3}}{2}\right) \\ \mathbf{s_4} &= \left(0.5, \frac{\sqrt{3}}{2}\right) \\ \mathbf{s_5} &= \left(0.5, -\frac{\sqrt{3}}{2}\right) \\ \mathbf{s_6} &= (-1,0) \\ \mathbf{s_7} &= (1,0) \end{aligned} \tag{4.9}$$

and where $\| \cdot \|$ is the Euclidean norm.

The examples of synaptic weight between neurons in value space layer are shown in Figure 4.5 Figure 4.6.

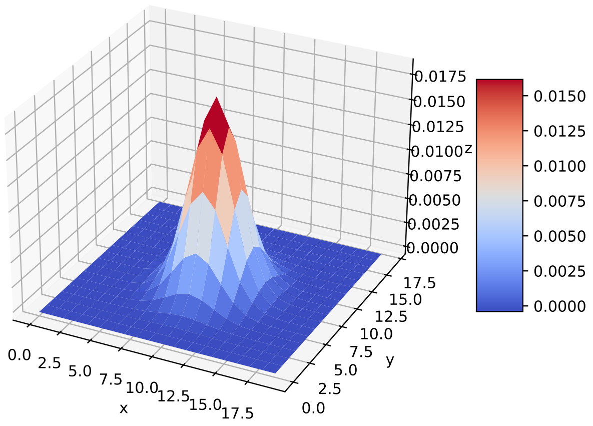

Figure 4.5.: Synaptic weight from cell (i_x, i_y) to cell $(9,8)$ in value space layer, with parameters: I=0.95, $\sigma = 0.13$, $T = 0.02N_x = 20$, $N_y = 18$

As shown in Figure 4.6, $(10,17)$, $(19,1)$ and $(0,0)$ are Neighboring. Because they have the biggest synaptic weight strength with (0,0). This situation also verifies the structure in Figure 4.4.

Because of the Mexican-hat connectivity Equation 4.4 and the twisted torus value space

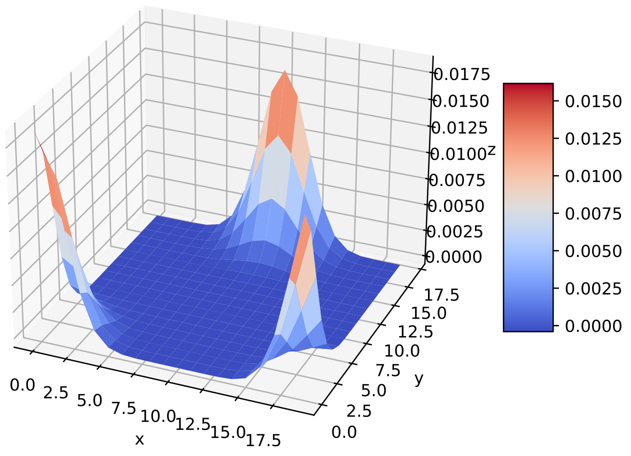

Figure 4.6.: Synaptic weight from cell (i_x, i_y) to cell $(0, 0)$ in value space layer, with parameters: I=0.95, $\sigma = 0.13$, $T = 0.02 N_x = 20$, $N_y = 18$

layer in Figure 4.3 can generate a stationary peak with the initial condition in subsection 4.2.7 without shift layer, which is shown in Figure 4.7.

4.2.3. Shift Layers

In the previous sections, we described the specific structure of the value space layer, the synaptic weight between neurons in it and the stable activity peak generated by synaptic weights in Equation 4.6. In this section we will introduce the structure of the shift layer. It is about the connecting synaptic weight from the value space layer to the shift layer, and the connecting synaptic weight from the shift layer to the value space.

The function of the shift layer is to move the peak Generated in the value space layer and as much as possible to keep the shape of the peak basically unchanged. Although in our model, the shape of the peak changes very slightly during rapid movement, but the shape change is generally symmetrical with respect to the center of the peak. So that this has almost no effect on the performance of GC model. The structure of the shift layer has the following characteristics:

- There are four shift layers corresponding to up, down, left, and right direction. Their function is to move the peak in the value space layer to their corresponding four directions.
- These four shift layers are parallel to the value space layer, and have the same size and the same number of cells.

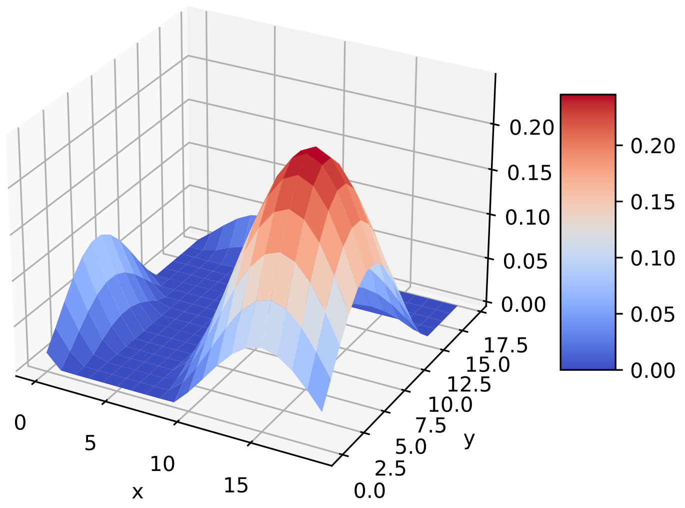

Figure 4.7.: Activity peak in value space layer without shift layers generated by synaptic weights from Equation 4.4 with parameters: I=0.95, $\sigma = 0.13$, $T = 0.02$, $T = 0.02 N_x = 20$ and by the inital conditionsubsection 4.2.7

- There is no connection between the shift layer and the shift layer. The shift layer is only connected to the value space layer.
- There is no connection between cells in the same shift layer.
- The neuron model in our shift layer is the same as that in the value space layer.

4.2.4. Synaptic Weights from Value Space Layer to Shift Layer

From the proof and description in next section section 4.4 we will know that we need the firing rates of the shift layer to be similar to the firing rates in value space layer. So we connect the cells of the value space layer to the shift layer with following synaptic weight

$$\begin{aligned} w_{ij}^{gl} &= p_l w_{ij} \\ w_{ij}^{gr} &= p_r w_{ij} \\ w_{ij}^{gu} &= p_u w_{ij} \\ w_{ij}^{gd} &= p_d w_{ij} \end{aligned} \tag{4.10}$$

Where w_{ij}^{gl},w_{ij}^{gr},w_{ij}^{gu} and w_{ij}^{gd} are synaptic weight from cell j in value space layer to cell i in shift left layer, shift right layer, shift up layer and shift down layer. w_{ij} is the synaptic weight from cell j in value space layer to cell i in value space layer. Because the shift layer and the value space layer have the same size and are parallel. Therefore the cell i in the shift layer and the cell i in the value space layer have the same position in their own layer. p_l, $p_r p_u$ and p_d are constant parameters. These parameters are equal in this model $p_l = p_r = p_u = p_d = p$.

4.2.5. Synaptic Weights from Shift Layer to Value Space Layer

In order to move the peak in the value space layer, we need to break the symmetry of the synaptic weights in value space layer as same as the most GC model [12][8]. But in the normal neuron system, we cannot directly change the synaptic weight to achieve the purpose of moving the peak. In some models, the cell itself carries the information of the direction. But this is not in line with our normal CANN model [8]. So we need additional neurons (cells in shift layers) to achieve the equivalent function of breaking the symmetry of the synaptic weights.

So that we define synaptic weights from shift layer to value space layer,which are shown in

Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11:

$$\begin{aligned}
w_{ij}^{rg} &= a_r \frac{I\left(\exp\left(\frac{\|c_i-c_j+D_x\|_{tri}^2}{\sigma^2}\right) - \exp\left(\frac{\|c_i-c_j\|_{tri}^2}{\sigma^2}\right)\right)}{dx} \\
w_{ij}^{lg} &= a_l \frac{I\left(\exp\left(\frac{\|c_i-c_j-D_x\|_{tri}^2}{\sigma^2}\right) - \exp\left(\frac{\|c_i-c_j\|_{tri}^2}{\sigma^2}\right)\right)}{dx} \\
w_{ij}^{ug} &= a_u \frac{I\left(\exp\left(\frac{\|c_i-c_j+D_y\|_{tri}^2}{\sigma^2}\right) - \exp\left(\frac{\|c_i-c_j\|_{tri}^2}{\sigma^2}\right)\right)}{dy} \\
w_{ij}^{dg} &= a_d \frac{I\left(\exp\left(\frac{\|c_i-c_j-D_y\|_{tri}^2}{\sigma^2}\right) - \exp\left(\frac{\|c_i-c_j\|_{tri}^2}{\sigma^2}\right)\right)}{dy}
\end{aligned} \tag{4.11}$$

Where w_{ij}^{rg}, w_{ij}^{lg}, w_{ij}^{ug} and w_{ij}^{dg} are synaptic weight from cell j in shift right layer, shift left layer,shift up layer and shift down layer to cell i in value space layer. D_x and D_y are constant vectors:

$$\begin{aligned}
D_x &= (d_x, 0) \\
D_y &= \left(0, d_y\right)
\end{aligned} \tag{4.12}$$

dx and dy is the constant parameters. c_i and c_j has been explained in Equation 4.7. a_r, a_l, a_u and a_d are constant parameters. In this model,they are equivalent.

In fact, w_{ij}^{rg}, w_{ij}^{lg}, w_{ij}^{ug} and w_{ij}^{dg} are Derivative of w_{ij} in x, $-x$, y and $-y$ direction. But because the derivative of arithmetic is difficult to calculate, we use approximate processing to get the derivative. But the impact of the approximate on the results is very small, when dx and dy are small enough.

It is these synaptic weights that the peak in the value space layer can move freely and basically does not change the shape of the peak. As for why I designed these synaptic weights from shift layer to value space layer in this way, I will explain in detail in the later part section 4.4.

4.2.6. Input Velocity into the Model

We have now built the value space layer and shift layers, and then we need to input the speed into the entire model to move the peak in the value space layer. From the neuron model Equation 4.1 we know that every cell i has external input I_i to receive external stimulus. In our model, we establish the relationship between speed and stimulus of cells in shift layers to

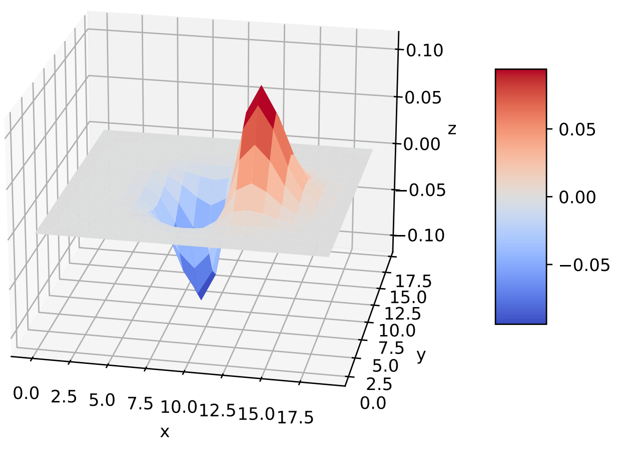

Figure 4.8.: Synaptic weights from cells in shift right layer to cell $(9, 8)$ in value space layer with I=0.95, $\sigma = 0.13$, $T = 0.02$ and d_x=0.1

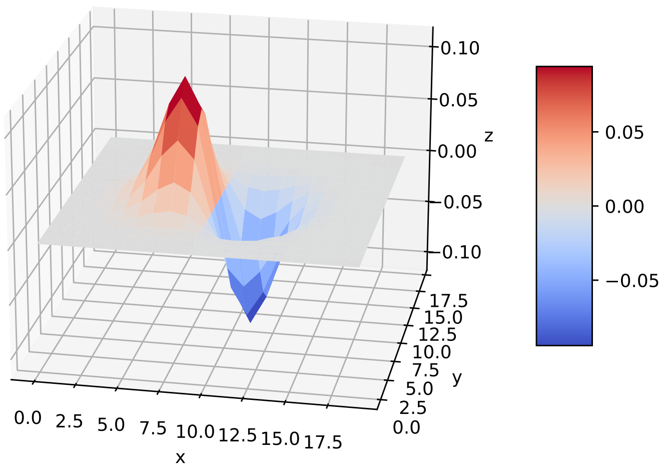

Figure 4.9.: Synaptic weights from cells in shift left layer to cell $(9, 8)$ in value space layer with I=0.95, $\sigma = 0.13$, $T = 0.02$ and d_x=0.1

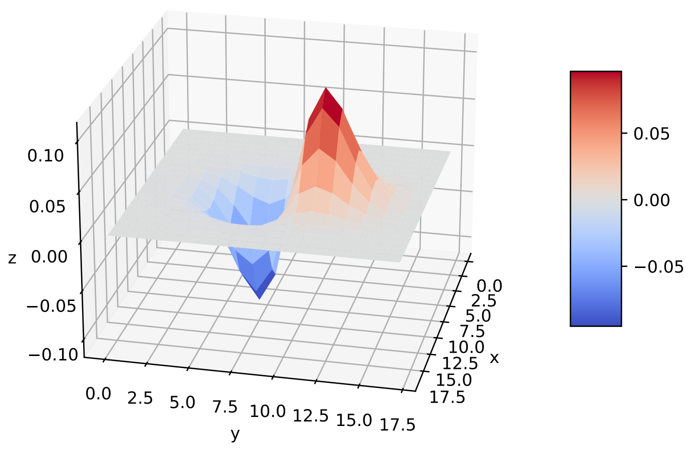

Figure 4.10.: Synaptic weights from cells in shift up layer to cell $(9, 8)$ in value space layer with I=0.95, $\sigma = 0.13$, $T = 0.02$ and d_y=0.1

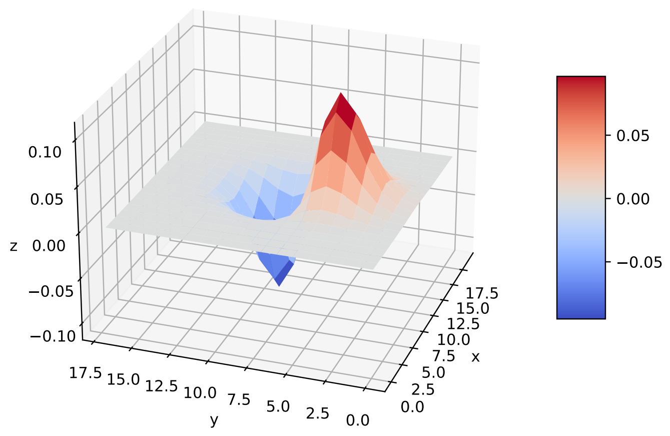

Figure 4.11.: Synaptic weights from cells in shift down layer to cell $(9, 8)$ in value space layer with I=0.95, $\sigma = 0.13$, $T = 0.02$ and d_y=0.1

achieve the purpose of the peak's move in value space layer. It is defined as:

$$\begin{aligned}
I_i^x(t) &= \begin{cases} w_v v_x^{in}(t) & v_x^{in}(t) > 0 \\ 0 & v_x^{in}(t) \leqslant 0 \end{cases} \\
I_i^{-x}(t) &= \begin{cases} 0 & v_x^{in}(t) > 0 \\ -w_v v_x^{in}(t) & v_x^{in}(t) \leqslant 0 \end{cases} \\
I_i^y(t) &= \begin{cases} w_v v_y^{in}(t) & v_y^{in}(t) > 0 \\ 0 & v_y^{in}(t) \leqslant 0 \end{cases} \\
I_i^{-y}(t) &= \begin{cases} 0 & v_y^{in}(t) > 0 \\ -w_v v_y^{in}(t) & v_y^{in}(t) \leqslant 0 \end{cases}
\end{aligned} \tag{4.13}$$

Where $I_i^x(t)$, $I_i^{-x}(t)$, $I_i^y(t)$ and $I_i^{-y}(t)$ are stimulus of cell i in shift right layer, shift left layer, shift up layer and shift down layer. W_v is a constant parameters and $W_v > 0$. $v_x^{in}(t)$ and $v_y^{in}(t)$ are input velocity in x and y direction in world coordinate system. As we found in the brain, there are GC in many parts, and their hexagonal structures have different spacing, rotation. So here we also need to introduce a rotation matrix to get different rotations in firing map:

$$\begin{aligned}
v^{in}(t) &= \begin{pmatrix} v_x^{in}(t) \\ v_y^{in}(t) \end{pmatrix} = R_s v(t) \\
&= \begin{pmatrix} \cos\epsilon & -\sin\epsilon \\ \sin\epsilon & \cos\epsilon \end{pmatrix} \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix}
\end{aligned} \tag{4.14}$$

Where $v_{in}(t)$ is the input velocity vector of GC model and $v(t)$ is the velocity vector of agent in world coordinate. ϵ is the angle of rotation of the hexagonal firing pattern. But based on the agent, we can only know the speed of the agent relative to its own frame. So we need the direction of the agent in world coordinate system, which is decoded from HDC, to get the speed of the agent in the world coordinate system. Therefore another rotation matrix is needed:

$$\begin{aligned}
v(t) &= R(t) v_a(t) \\
&= \begin{pmatrix} \cos(\theta(t)) & -\sin(\theta(t)) \\ \sin(\theta(t)) & \cos(\theta(t)) \end{pmatrix} v_a(t)
\end{aligned} \tag{4.15}$$

where $v(t)$ is the velocity of agent in world coordinate, $v_a(t)$ is the velocity vector of agent in egocentric coordinate system, $R(t)$ is the rotation matrix and $\theta(t)$ is the direction of the agent in world coordinate.

That is to say, in which direction the agent moves, a positive stimulus is given to which shift layer. When we give a positive stimulus to which shift layer, the peak in the value space layer will move in which direction.

4.2.7. Initializing the Grid Cells Model

After building a complete system structure, we need to initialize the entire network to generate a peak in value space layer. The firing rate (A) of neurons in every layers are initialized with a random activity that is uniformly distributed between 0 and $\frac{1}{\sqrt{N}}$, where N is the total number of neurons in neuron's layer. The initialization of the synaptic currents B of neurons is also involved in our model. The method is the same as the initialization of activity of neuron: uniformly between 0 and $\frac{1}{\sqrt{N}}$. Next is our specific algorithm for initializing the network:

Algorithm 1: GC Network Initialization

```
Input :
        $t_{settle}$: settling time
        $t_{interval}$: time interval to ensure stability
        $\epsilon$: maximum total change during $t_{interval}$
set the initial random activity $A$ and the synaptic currents $B$ of cells in every five layers
  between 0 and $\frac{1}{\sqrt{N}}$;
run simulation for $t_{settle}$;
$\delta = \infty$;
while $\delta > \epsilon$ do
    $\delta = 0$;
    for $t$=0; $t$<$t_{interval}$; $t$=$t + dt$ do
        save firing rates in $R_{old}$;
        run simulation for time $dt$;
        save firing rates in $R_{new}$;
        $\delta = \delta + \sum_{i=0}^{n-1} |R_{old}[i] - R_{new}[i]|$;
    end
end
```

After the algorithm 1 with parameters in Table 4.1, we can get a stable peak in the value space layer, as shown in Figure 4.12. Our model show that, the whole structure is very friendly to the initial conditions.

Table 4.1.: Initialization Parameters

t_settle	50	ms
$t_interval$	50	ms
dt	0.5	ms
ϵ	0.001	Hz

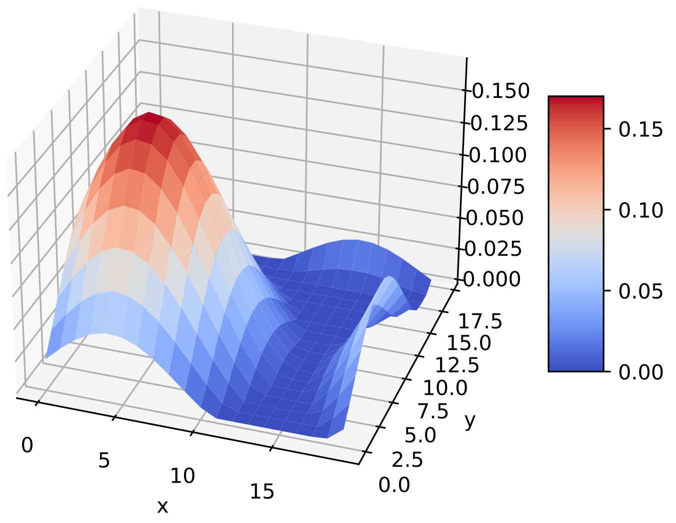

Figure 4.12.: Activity peak generated with synaptic weights from Equation 4.4 by I=0.95, $\sigma = 0.13$, $T = 0.02$ and with shift layers

4.3. Decoding of Value Space

The advantage of our GC model based on CANN lies in its ability to use the activity of cell populations to resist noise. If we only use the highest point of the peak as the position of the peak in the value space layer, not only our accuracy will be greatly reduced, but also the anti-noise performance will also be reduced. Because the structure of the value space layer of the GC is a 3-dimensional twisted ring structure shown in Figure 4.3, we decided to use **mean of circular quantities** to calculate the peak position. First of all, each neuron corresponds to one position in the 3-dimensional twisted ring structure:

$$\begin{aligned} \theta_i^x &= \frac{2\pi i_x}{N_x} \\ \theta_i^y &= \frac{2\pi i_y}{N_y} \end{aligned} \tag{4.16}$$

Where θ_i^x and θ_i^y is the position of cell i in x and y direction ring, i_x and i_y are the column and the row number of cell i in grid sheet. A vector v_i is generated for every neuron i in value space layer:

$$v_i(t) = \begin{pmatrix} v_i^x(t) \\ v_i^y(t) \end{pmatrix} = \begin{pmatrix} \frac{A_i(t)\cos\theta_i}{S(t)} \\ \frac{A_i(t)\sin\theta_i}{S(t)} \end{pmatrix} \tag{4.17}$$

Where S is the sum of firing rate of neurons in value space layer:

$$S(t) = \sum_{i=1}^{N} A_i(t) \tag{4.18}$$

the decoded position γ is simply obtained by using the 2-argument arctangent *arctan*2:

$$\gamma(t) = \begin{pmatrix} \frac{arctan2(\sum_{i=1}^{N} v_i^x(t))}{2\pi} \\ \frac{arctan2(\sum_{i=1}^{N} v_i^y(t))}{2\pi} \end{pmatrix} \tag{4.19}$$

We can calculate the movement speed of peak based on the change of the decoded peak's position $\gamma(t)$. At the same time, path integration can be calculated according to the moving speed of the decoded position.

4.4. Theoretical Explanation of Shift Layer

In previous research, we found that if we want to move the peak in the value space layer, we must break the symmetry of the synaptic weights between the cells in value space layer. The same method of moving peaks is used in most GC models: [12][8][55][98].For example, as mentioned in [12]:

$$w_{ij}^{shift}(t) = I \exp\left(-\frac{\| c_i - c_j + \alpha R_\beta v(t) \|_{tri}^2}{\sigma^2}\right) - T \tag{4.20}$$

, where $w_{ij}^{shift}(t)$ is the synaptic weights between neurons in value space layer without shift layers, so that the peak in this layer can move in the direction to $\alpha R_\beta v(t)$ with the speed corresponding to $v(t)$. Except for α R_β, the other parameters are the same as in Equation 4.4. $v(t) = (v_x, v_y)$ is the speed vector, which represents the speed of the agent in environment as same as the $v(t)$ in Equation 4.14. α is a scale constant parameters corresponding to the parameter w_v in Equation 4.13. R_β is the rotation matrix corresponding to the rotation matrix R_s in Equation 4.14. This input does not depend on any information about the location of the agent, only depend on the agent's direction decoded from HDC and the velocity of the agent.

It is possible to increase or decrease the spacing of the hexagonal firing pattern shown in Figure 2.2, as well as to rotate the hexagonal firing pattern, by changing only two parameters in the model: the gain $\alpha \in \mathbb{R}_+$ and the rotation angular $\epsilon \in [0, \pi/3]$. The input of the network is thus modulated and biased by these gain and rotation parameters.

In fact, this is just a movement of the synaptic weights in the direction of $\alpha R_\beta v(t)$. So when we use Taylor's first-order expansion, we can get:

$$w_{ij}^{shift}(t) \approx w_{ij}(t) + \gamma(v(t))\vec{n} \cdot \bigtriangledown w_{ij}(t) \tag{4.21}$$

Where $w_{ij}^{shift}(t)$ is the synaptic weights in Equation 4.20, $w_{ij}(t)$ is the static synaptic weights in Equation 4.6, $\vec{n} \cdot \bigtriangledown$ is the directional derivative in the direction of an arbitrary unit vector $\vec{n}$ which is the direction of velocity $R_\beta v(t)$ and $\gamma(v(t))$ is a parameter depending on the norm of velocity. if we use the new $w_{ij}^{shift}(t)$ in Equation 4.20 to shift the peak in value space layer, considering the neuron model Equation 4.1 and without shift layers:

$$\begin{aligned} B_i(t+1) &= \sum_{j=1}^{N} A_j(t) w_{ij}^{shift} \\ &\approx \sum_{j=1}^{N} A_j(t)(w_{ij}(t) + \gamma(v(t))\vec{n} \cdot \bigtriangledown w_{ij}(t)) \\ &= \sum_{j=1}^{N} A_j(t) w_{ij}(t) + \underbrace{\sum_{j=1}^{N} A_j(t)\gamma(v(t))\vec{n} \cdot \bigtriangledown w_{ij}(t)}_{I_i} \end{aligned} \tag{4.22}$$

, the GC model in this thesis, the external input I_i in Equation 4.22 is from shift layers.

First, because of the connection from value space layer to shift layers, the firing rate of neurons in shift layers is similar to the firing rate of neurons in value space layer and the shift layers will also be stimulated by speed, we can infer the following conclusions:

$$\begin{aligned} b_r(v_x(t))A_i(t) &\approx A_i^r(t) \\ b_l(v_{-x}(t))A_i(t) &\approx A_i^l(t) \\ b_u(v_y(t))A_i(t) &\approx A_i^u(t) \\ b_d(v_{-y}(t))A_i(t) &\approx A_i^d(t) \end{aligned} \tag{4.23}$$

Where $A_i(t)$ is the firing rate of neuron i in value space layer, and A_i^r, A_i^l, A_i^u and A_i^d are firing rate of neuron i in shift right, left, up, down layers. b_r, b_l, b_u and b_d are variable only depend on the corresponding velocity of the own shift layer. Now we bring these conditions into Equation 4.1. The following is the expression of cell firing rate in the value space layer with shift layers:

$$\begin{aligned}
B_i(t+1) &= \sum_{j=1}^{N} A_j(t)w_{ij} + I_i \\
&= \sum_{j=1}^{N} A_j(t)w_{ij} + \sum_{j=1}^{N} A_j(t)^r w_{ij}^{rg} + \sum_{j=1}^{N} A_j(t)^l w_{ij}^{lg} + \sum_{j=1}^{N} A_j(t)^u w_{ij}^{ug} + \sum_{j=1}^{N} A_j^d(t) w_{ij}^{dg} \\
&\approx \sum_{j=1}^{N} A_j(t)w_{ij} + \sum_{j=1}^{N} b_r(v_x(t))A_j(t)w_{ij}^{rg} + \sum_{j=1}^{N} b_l(v_{-x}(t))A_j(t)w_{ij}^{lg} \\
&+ \sum_{j=1}^{N} b_u(v_y(t))A_j(t)w_{ij}^{ug} + \sum_{j=1}^{N} b_d(v_{-y}(t))A_j(t)w_{ij}^{dg}
\end{aligned} \tag{4.24}$$

It is not difficult for us to find that the corresponding synaptic weight w_{ij}^{rg}, w_{ij}^{lg}, w_{ij}^{ug} and w_{ij}^{dg} in Equation 4.11 are the derivative of w_{ij} in corresponding direction, so we bring these conditions back into the neuron model Equation 4.1 and Equation 4.24:

$$\begin{aligned}
B_i(t) = \underbrace{\sum_{j=1}^{N} A_j(t)w_{ij}}_{\text{stable part}} + \sum_{j=1}^{N}(b_r(v_x(t)) - b_l(v_{-x}(t)))A_j(t)\vec{n}_x \bigtriangledown w_{ij} \\
+ \sum_{j=1}^{N}(b_u(v_y(t)) - b_d(v_{-y}(t)))A_j(t)\vec{n}_y \bigtriangledown w_{ij}
\end{aligned} \tag{4.25}$$

, where $\vec{n}_x$ and $\vec{n}_y$ is are unit vectors in x and y direction.

So when our agent need to be stationary in the environment, we can know that $b_r(v_x(t)) = b_l(v_{-x}(t))$ and $b_u(v_y(t)) = b_d(v_{-y}(t))$. So that the Equation 4.24 only remain the "stable part" and the peak in the value space layer will Keep still. Equation 4.22 compared with Equation 4.24, the have the same form and function. I will take the agent moving in the x direction as an example. The move in Equation 4.22 is:

$$B_i(t+1) = \sum_{j=1}^{N} A_j(t)w_{ij}(t) + \sum_{j=1}^{N} A_j(t)\gamma(t)\vec{n}_x \cdot \bigtriangledown w_{ij} \tag{4.26}$$

The move in Equation 4.24 is:

$$\begin{aligned}
B_i(t+1) &= \sum_{j=1}^{N} A_j(t)w_{ij}(t) + \sum_{j=1}^{N} \underbrace{(b_r(v_x(t)) - b_l(v_{-x}(t)))}_{\gamma(v(t))} A_j(t)\vec{n}_x \bigtriangledown w_{ij} \\
&= \sum_{j=1}^{N} A_j(t)w_{ij}(t) + \sum_{j=1}^{N} A_j(t)\gamma(v(t))\vec{n}_x \cdot \bigtriangledown w_{ij}
\end{aligned} \tag{4.27}$$

Under our design ideas, we need to adjust the parameters to achieve the good performance of the entire model. One of the important performance indicators is about the linear relationship between the speed of our input and the speed of peak movement. And this performance is related to $b_r(v_x(t))$, $b_r(v_{-x}(t))$, $b_r(v_{-y}(t))$ and $b_r(v_y(t))$. If they have a linear relationship with the corresponding speed, then our input speed and peak movement speed will also show a linear relationship. In chapter 5 we will evaluate the performance of our model.

4.5. Grid Cells to Place Cells

From Figure 3.2 we know that hippocampal PC are thought to build their place fields mainly by converting the many-location responses of GC into firing that is usually restricted to a single location. Understanding the rules and governing this transformation will be an important step in understanding the calculation method and function of the HC. It has been proposed that place fields can be generated from GC inputs by a simple summation and threshold operation. That is, if a PC receives input from an arbitrary set of GC, the activation of the PC can be prevented everywhere except in the single region where the input to the cell is maximal [99]. We can know from previous neurological research [100] that there are GC with different intervertex spacing, rotation angles and phase in hexagonal firing pattern in Figure 2.2 in the brain. However, if the geometric alignment of the input grid is not restricted, different subsets of the input will almost always produce similar levels of synaptic excitation in multiple areas covering most of the environment [7]. A simple threshold mechanism would not be able to single out one of these regions [7].

As shown in Figure 4.14, the structure of PC is that: the PC connects every GC with the synaptic weight,which is trained by Hebbian learning. The neuron model of our PC model still uses the neuron model in Equation 4.1. But this PC has only one layer and cells are not connected to each other, only different GC are connected to every PC. For PC decoding, we implement which PC has the largest firing rate, then it means which PC is firing:

$$P_{index} = \arg\max_{j} A_j^p \tag{4.28}$$

Where P_{index} is the index of firing PC, It also indicates which area the PC model predict the agent is in. A_j^p is the firing rate of cell j in PC. Our model is still different from the real neuron system. Because we have countless GC in the brain. But our model only has a limited GC. So we need to chose suitable GC with different phase, spacing and orientation according to the size of the environment and the shape of the environment, the number of PC and etc. Our choice of GC determines the performance of our PC. A subset of grids with overlapping vertices at a single location could be optimally selected from randomly aligned grid inputs by choosing a suitable synaptic weight vector, e.g., via Fourier analysis [7], a fitting algorithm [101], or independent component analysis [102]. It is not known, however, how this task can be autonomously accomplished at a behaviorally relevant time scale with physiological mechanisms. In the model in this article, we investigated a Hebbian learning rule from subsubsection 3.5.1.2 within a minimal, rate-based network model of a layer of presynaptic GC and modifiable connections onto an postsynaptic place layer as mentioned in subsection 3.5.1 and [89]:

$$\frac{dw_{ij}^{gp}}{dt} = k(\overline{x_j} - \theta_p)\overline{x_i} \tag{4.29}$$

, where w_{ij}^{gp} is the synaptic weight from GC j to PC i, $\overline{x}_i(t)$ and $\overline{x}_j(t)$ are the firing rates of the PC i and GC j, k is the learning rate factor and θ_p is a threshold on the presynaptic GC firing rate. This model does not intend to imitate the connection of the real GC to the PC, because the real connection is far more complicated than this model. This model just provides an idea to solve PC problems.

The condition for terminating learning lies in the processing of the predicted position from GC and the real position of agent. If the area corresponding to our firing PC is exactly the same as the agent's location in the environment, this prediction is successful. If we make 1000 consecutive predictions, then we stop updating the weight.

4.5.1. Encoding of Place Cells Firing Rate

In the training process, we need to encode the PC from the agent's position in the environment. We divide the environment into different areas such as Figure 4.13. Each area corresponds to a PC. When our agent is in this area, we encode the firing rate of the PC corresponding to the area as 1, and encode other PC as 0:

$$A_i^p(t) = \begin{cases} 1 & i = P_{index} \\ 0 & \text{others} \end{cases} \tag{4.30}$$

, where $A_i^p(t)$ is the firing rate of PC i at time t. For example, if our agent is in zone 1. Then the firing rate of the PC corresponding to our 1 area is 1, and the firing rate of the other PC is 0.

4.5.2. Grid Cells Firing Rate

Our GC can change the spacing and rotation of the hexagonal firing pattern by adjusting w_v and R_s in Equation 4.14 and Equation 4.13. The phase of the hexagonal firing pattern is achieved through our random initial firing rate of gird cells. But today's computers are not suitable for simulating a large number of GC, but the hardware more suitable for our model has not been developed yet, which is shown in subsubsection 3.1.3.3. Although it can be seen from the following results that our GC has very good performance. So we need to use the ideal GC from [102] model to form the connection mechanism from our GC to the PC. The rate map of GC over $(\vec{x})$ can be described by a sum of three 2-d sinusoids:

$$g_j(\vec{x}) = \frac{1}{4.5}\left(\sum_{\gamma=1}^{3} \cos 2\pi \vec{u}_j^{\gamma} \frac{\vec{x} - \vec{\zeta}_j}{a_j} + 1.5\right) \tag{4.31}$$

, where $g_j(\vec{x})$ is the firing rate of gird cell j at position $\vec{x}$, $\vec{\zeta}_j$ is the spatial phase or offset of GC j, and a_j is the grid spacing. The $\vec{u}_j^{\gamma}$ are direction vectors that are orthogonal to the main

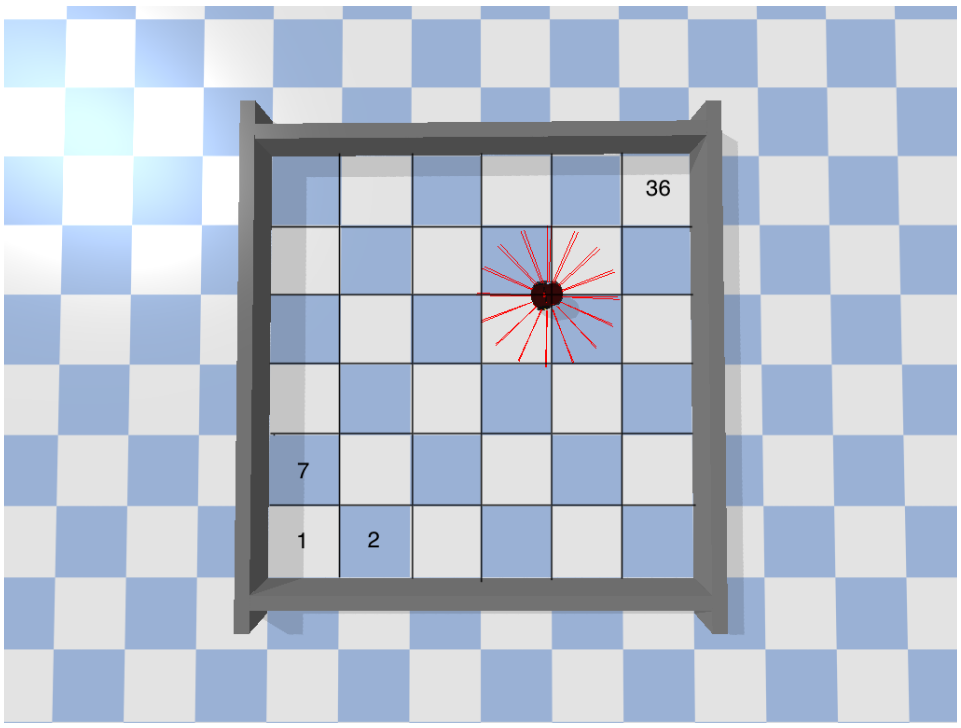

Figure 4.13.: In a 6 × 6 scene, we divide the entire scene into 36 areas. Each area corresponds to a PC.

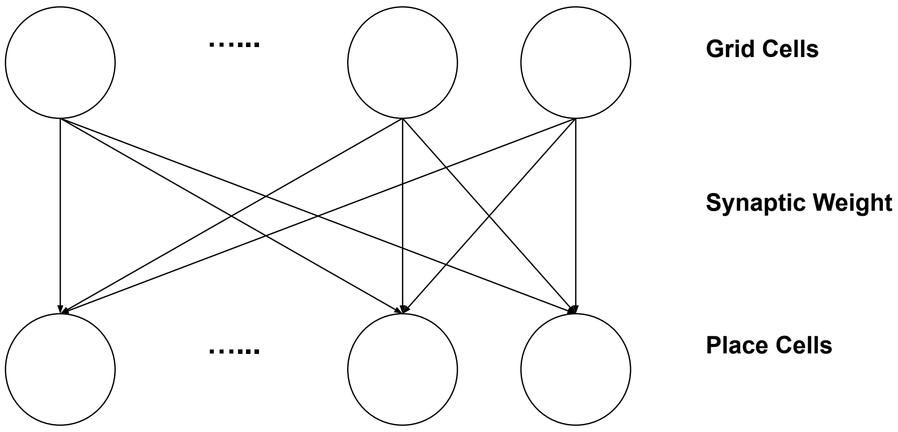

Figure 4.14.: The structure of PC in our model

axes of the hexagonal firing pattern:

$$\vec{u}_j^1 = \frac{2}{\sqrt{3}} \begin{pmatrix} \cos\left(\phi_j + \frac{\pi}{6}\right) \\ \sin\left(\phi_j + \frac{\pi}{6}\right) \end{pmatrix}$$
$$\vec{u}_j^2 = \frac{2}{\sqrt{3}} \begin{pmatrix} \cos\left(\phi_j + \frac{\pi}{2}\right) \\ \sin\left(\phi_j + \frac{\pi}{2}\right) \end{pmatrix} \qquad (4.32)$$
$$\vec{u}_j^3 = \frac{2}{\sqrt{3}} \begin{pmatrix} \cos\left(\phi_j + \frac{5\pi}{6}\right) \\ \sin\left(\phi_j + \frac{5\pi}{6}\right) \end{pmatrix}$$

, where ϕ_j represents the orientation of hexagonal firing pattern. An example of a GC firing map is shown in Figure 4.15. In the model of this article, we will adjust these ϕ_j, a_j and

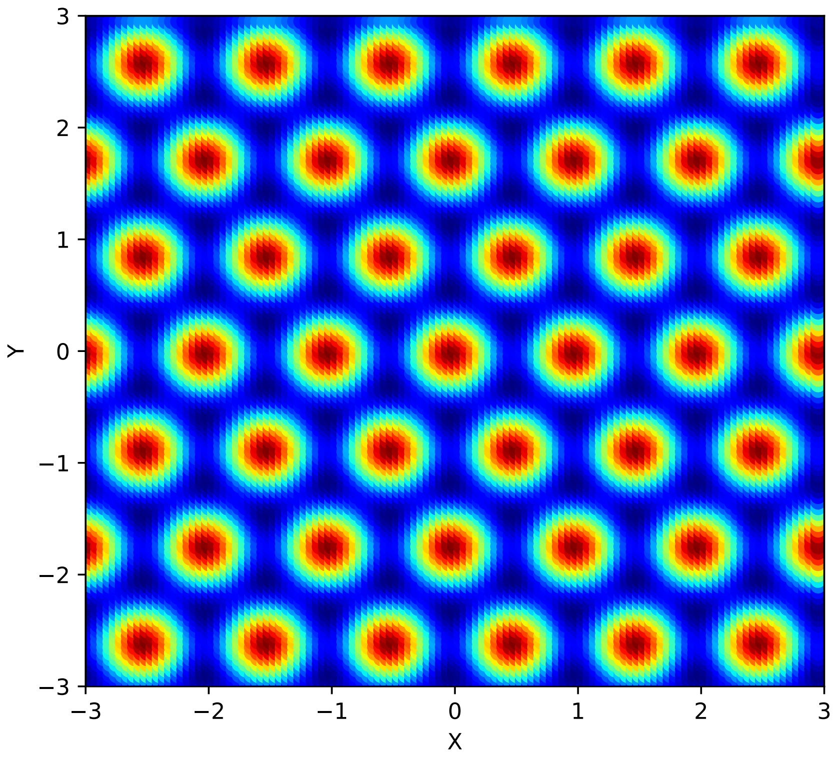

Figure 4.15.: Schematic of GC firing rate map with $\phi_j = 0$, $a_j = 1$ and $\zeta_j = (0, 0)$. As in all firing rate and activation maps in this article, red indicates maximum, while blue denotes zero.

ζ_j parameters according to the number of PC and the size of the environment to generate different GC.

The whole algorithm with GC firing rate in subsection 4.5.2 and the PC firing rate in subsection 4.5.1 is shown in algorithm 2.

Algorithm 2: Algorithm of Hebbian Learning

```
Input :
        $p(t)$: position of the agent
        $A_g(t) \in \mathbb{R}^{n_g}$: encoded GC firing rate vector
        $\epsilon$: maximum successful steps
        $\theta_p$: threshold on the presynaptic GC
        $dt$: time step for simulation
        $t_{\max}$: max simulation time
$w = zeros(n_p, n_g);$
$counter = 0;$
while $counter < \epsilon$ do
    for $t=0; t<t_{max}; t=t+dt$ do
        calculate the area $p_r(t)$ the agent in, according to position $p(t)$;
        encode the $A_p(t) \in \mathbb{R}^{n_p}$: encoded PC firing rate vector according to $p_r(t)$;
        for $i=0; i < n_p; i = i+1$ do
            for $j=0; j < n_g; j = j+1$ do
                $w[i][j] = w[i][j] + dt(A_g[j] - \theta_p)A_p[j];$
            end
        end
        calculate the predicted PC firing rate vector $A_p^e(t)$;
        decode the $A_p^e(t)$ to get predicted area $p_e(t)$ ;
        if $p_r(t) == p_e(t)$ then
            counter$+ = 1$;
        end
        else
            counter$= 0$;
        end
    end
end
```

5. Results and Discussions

In this chapter, we will run the robot (Pioneer P3DX) in a simulation environment to obtain its position and velocity information to drive the GC model and test its performance. Following, we will provide details on the simulation environment, physics engine, model parameters and We will also analyze the accuracy and the biological plausibility of the GC model and PC model. From the overall results we can come to this conclusion that the GC model has high accuracy with small size and it can be used as a viable component of an biologically inspired robotic navigation system. At the same time, through Hebbian learning, PC also shows good performance that it can show the specific location of the agent in the environment with enough GCs as input.

5.1. Simulation Environment

5.1.1. Simulation Physics Engine

We use the Pybullet [103] as the simulation physics engine. Pybullet is a Python module that can be used for physical simulation of robots, games, visual effects and machine learning. Using pybullet, we can load robot description files in URDF, SDF, MJCF and other file formats. Pybullet provides forward dynamic simulation, inverse dynamic calculation, forward and inverse kinematics, collision detection and ray intersection query. The Bullet Physics SDK also provides many examples of Pybullet robots, such as the simulated quadruped robot Minitaur, the simulated human running, which uses tensorflow to make decisions, and KUKA robot. In addition to physical simulation, it also has rendering bindings, including CPU render (TinyRenderer) and OpenGL visualization, and supports virtual reality such as HTC Vive and Oculus Rift. Pybullet also has the ability to perform collision detection queries (nearest points, overlapping pairs, ray intersection tests, etc.) and add debug rendering (debug lines and text). Pybullet has a cross-platform built-in client server, supports shared memory, UDP and TCP networks. This physics engine is not only very lightweight, but also we can obtain enough information of the robot and also can accurately control the robot. At the same time it is very compatible with python. Because of these advantages we chose Pybullet as the physics engine.

5.1.2. Simulation Agent

The Pioneer P3DX robot provided by [104] is a model of a mobile robot implementing differential drive with a multitude of sensors in Figure 5.1. For this simulation 16 ultrasonic ray-based proximity sensors are added to the robot at about 15cm height and distributed around the circular body with an angular spacing of approximately 22.5° around the Z axis, which is shown in Figure 5.2. The velocity of Pioneer P3DX is from the function *getBaseVelocity* in Pybullet. In a real world experiment, an inertial measurement unit can be used for sensing linear velocity. The robot is controlled by setting left and right wheel speeds.

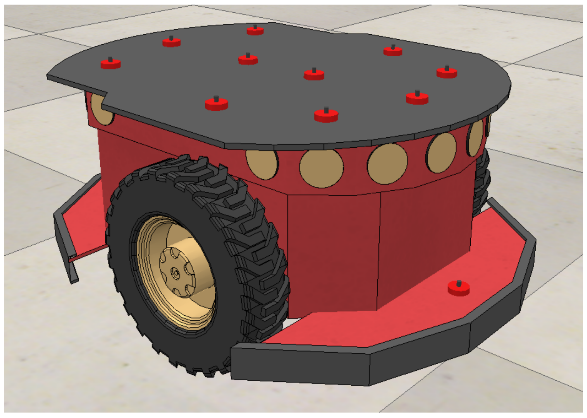

Figure 5.1.: Pioneer P3DX robot model — The model implements differential drive and is equipped with 16 ultrasonic, ray-based proximity sensors.

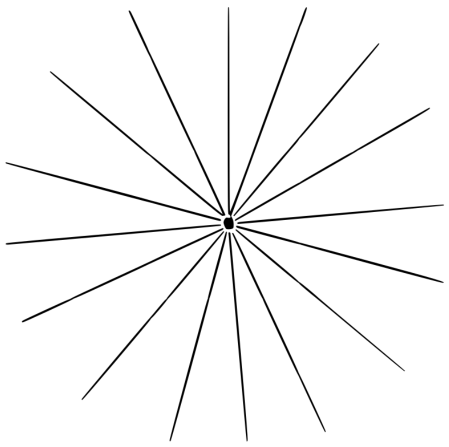

Figure 5.2.: Ultrasonic, ray-based proximity sensor configuration — The 16 sensors are spaced at 22.5 degrees around the z axis at 20 cm height.

5.1.3. Simulation Scenes

The whole simulation scene is shown in the Figure 5.3. The initial position of the robot is in the middle of the square area. These walls limit the robot's range of movement.

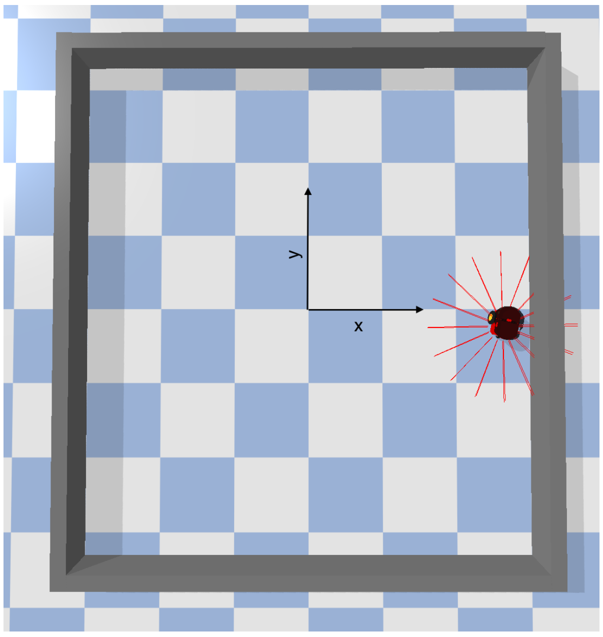

Figure 5.3.: The scenes of the simulation: the environment is a square area of 6 meters by 6 meters separated by walls. The center of the area is the origin. The horizontal direction is the x axis and the vertical direction is the y axis. The range of the area is $x \in [-3,3]$ and $y \in [-3,3]$.

5.1.4. Random Walk

The initial position of the robot is in the middle of the square area, which is the position of the origin. Then we control the speed of the two wheels of the robot so that the robot can walk randomly in the square area to get position and velocity of the robot while avoiding obstacles. The obstacle avoidance algorithm we used is based on Braitenberg vehicle.

A Braitenberg vehicle is an agent that can autonomously move based on its sensor inputs. It has primitive sensors that measure some stimulus at a point, and wheels (each driven by its own motor) that function as actuators or effectors. In the simplest configuration, a sensor is directly connected to an effector, so that a sensed signal immediately produces a movement of the wheel. Depending on how sensors and wheels are connected, the vehicle exhibits different behaviors (which can be goal-oriented). This means that, depending on the sensor-motor wiring, it appears to strive to achieve certain situations and to avoid some situations [105]. For example, the connections between sensors and actuators for the simplest braitenberg vehicles as shown in [106] Figure 5.4 (**2a and 2b**), they will lead to different results. For the **2a**: this agent has two (left and right) symmetric sensors (e.g. light detectors) each stimulating a wheel on the same side of the body. This car represents the animal's escape from light. It obeys the following rule: more light right $\rightarrow$ right wheel turns faster $\rightarrow$ turns towards the left $\rightarrow$ away from the light. This is efficient as a behavior to escape from the light source, since the vehicle can move in different directions, and tends to orient towards the direction from which least light comes. And for the **2b**: The agent has the same two (left and right) symmetric sensors (e.g. light detectors), but each one stimulating a wheel on the other side of the body. It obeys the following rule: More light left $\rightarrow$ right wheel turns faster $\rightarrow$ turns towards the left $\rightarrow$ closer to the light. As a result, the vehicle follows the light and it moves to be closer to the light.

These correspond to biological positive and negative taxes [106] present in many animals species. For our model, we use the distance information between the obstacle and the robot from 16 ray-based proximity sensor as shown in Figure 5.2 as the input. Our left and right wheels both have a parameter vector, and each vector is multiplied by the distance information vector, which consists of distance information from ultrasonic ray-based proximity sensors, to control the speed of each wheel. Our algorithm is similar to 2*a* in Figure 5.4. For example, when the robot approaches the obstacle on his right, the wheel on the right will gain a greater speed to achieve the purpose of staying away from the obstacle.

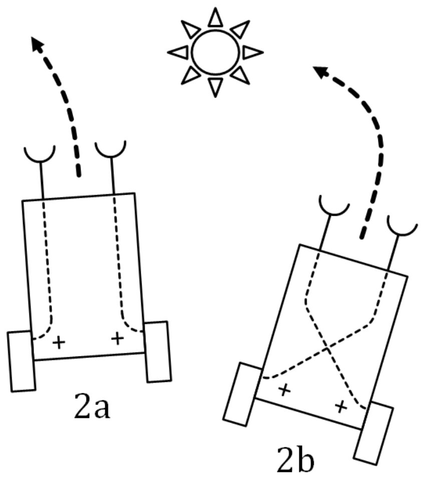

Figure 5.4.: A simple model of Braitenberg Vehicle [106]

5.2. Performance of the GC

In this section, we will show the performance of our GC model through different indicators. At the same time, we will show the influence of some important parameters on the performance of our GC model. After we systematically compare the influence of parameters on model performance, we have got the following parameters in Table 5.1 to achieve relatively good performance. In following parts, we will show the impact of some parameters on the performance of our GC model and the performance of our model under various indicators with parameters in Table 5.1.

5.2.1. Stability of the Model in a Static State

First, our model needs to be stable enough when there is no external input to our GC model,it means that the peak in value space layer of our GC model will not drift and stay stationary when the agent is stationary. in other words, the activity of neurons in value space layer should remain unchanged over time. This is an important indicator for performance of GC model. We judge whether the network is stable by calculating the sum of the absolute difference between the firing rate of the cells in the value space layer at this moment and the firing rate at the previous moment without input:

$$\delta(t) = \sum_{i=0}^{N-1} |A_i(t) - A_i(t-1)| \tag{5.1}$$

, Where $\delta(t)$ is the static error, A_i is the firing rate of cell i in value space layer and N is the number of neurons.

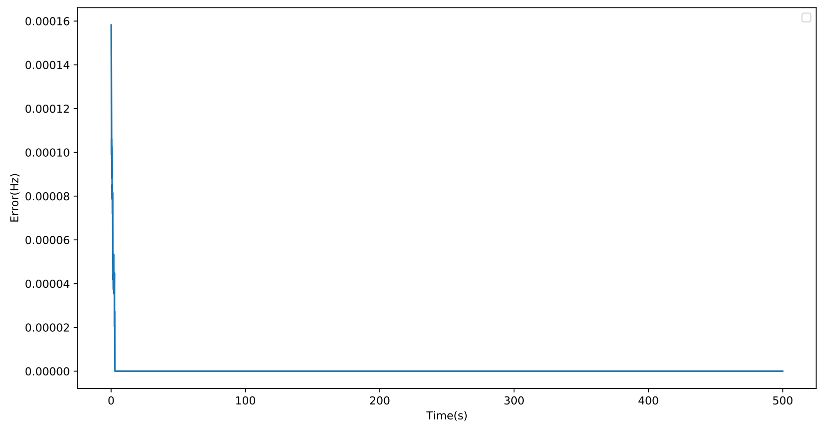

Figure 5.5.: Static error Equation 5.2.1 of the neuron without input with parameters in Table 5.1

As shown in Figure 5.5, there is still a certain error of this GC model in the first few milliseconds because of the initialized settings parameters ϵ in algorithm 1. But immediately our error tends to 0. It shows that as time goes by, the network tends to become static. Therefore the network is very stable when the model has no inputs because of reasonable synaptic weights and structure.

5.2.2. Stability under Fixed Input

In addition to keeping the peak in the value space layer static when the agent is static, the peak also need to move at a constant speed, when we have a constant and continuous input to the GC model. If the peak's moving speed fluctuates Violently asymmetrically or fluctuates in a big range, when the input of our model is a fixed value, then the accuracy of the GC will be greatly affected.

Table 5.1.: Final Parameters for GC Model

a_r	0.02	None
a_l	0.02	None
a_u	0.02	None
a_d	0.02	None
N_x	20	None
N_y	18	None
p_r	1	None
p_l	1	None
p_u	1	None
p_d	1	None
I	0.95	None
σ	0.13	None
d_x	0.1	None
d_y	0.1	None
T	0.02	None

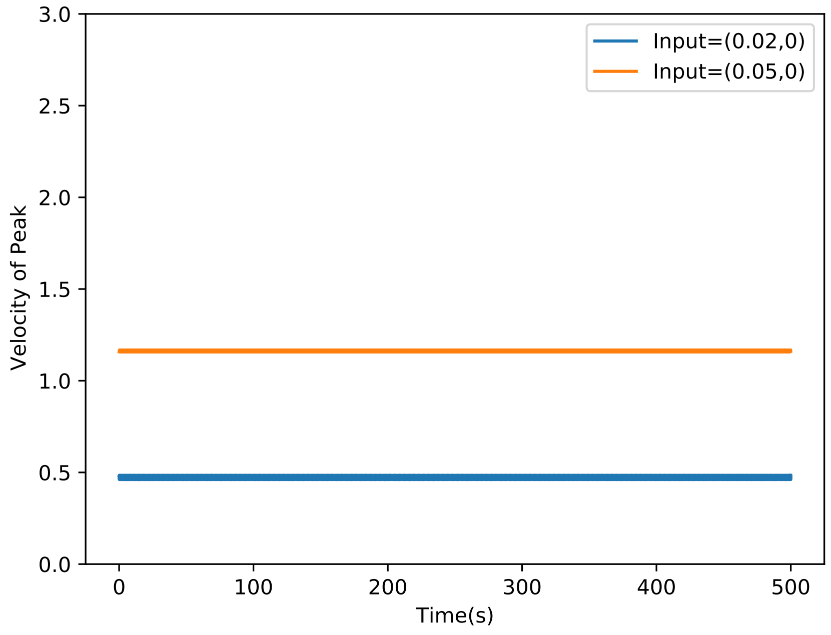

Figure 5.6.: velocity of peak, which is decoded from Equation 4.19, with constant input $I_i^x(t) = 0.05$ and $I_i^x(t) = 0.02$ with parameters in Table 5.1

As shown in Figure 5.6, our peak's moving speed symmetrically fluctuates in a very small range around a constant. And we use the least square method to calculate the linear relationship between time T and speed. Our time-related parameters are: 2.84×10^{-9} and -5.035×10^{-8}, they are very small. It means that there is no relationship between moving speed of the peak and time. From the above results, we can draw the following conclusion: the speed of our peak in value space layer is very stable under a fixed input. That is to say, when we perform path integration with constant inputs, there will be no big drift.

5.2.3. Linearity between Stimulus and Peak Velocity

Except the stability of the GC model, It is very important for the performance of the GC that the movement speed of the peak should to be linearly proportional to our input stimulus and it will determine the accuracy of the GC model. Once the relationship between inputs and moving speed of the peak in value space layer is nonlinear, then the hexagonal pattern will not be formed and the path integration, which is based on GC model, will be inaccurate. After our analysis and experiments, we found that the main parameter that affects linearity and the range of linear space is the synaptic weights from the value space layer to the shift layer. Therefore, we mainly change p_r, p_l, p_u and p_d to see their impact on performance. In order to form equilateral triangles in the hexagonal pattern as much as possible, we will make the composition of the network more symmetrical. So in our model$p_r = p_l = p_u = p_d = p$.

We gradually increase the input with different parameters p, so as to observe the relationship between movement speed of peak in value space layer and different parameters with different input stimulus as shown in Figure 5.7. From Figure 5.7 we can draw the following conclusions:

- First, as the input stimulus continues to increase, the velocity of peak in value space layer increases more and more slowly. The main reason is normalization. Because the model has a normalization mechanism, when the GC model have a relatively large input stimulus, the firing rate of the neuron in shift layers will gradually rise more and more slowly. But this does not limit the input speed of the GC model. Because we can adjust the parameter (w_v) between input speed and stimulus to compressed the input speed to a linear space in input stimulus.
- We can also see that as the parameters p increase, the range of linear space between input stimulus and the speed of peak also increase. In fact, this linear space is determined by the linearity between b_r, b_l, b_u, b_d in Equation 4.4 and velocity input to the GC model. If these parameters about $v(t)$ are linear, then relationship between moving speed of the peak in the value space layer and the velocity inputs are also linear. The biggest linear space occurs mainly when p approaches 1, the firing rate of cells in the shift layer will get more and more similar to the firing rate of cells in value space layer. So when we increase the input to neurons in shift layer, the change of it's firing rate is small compared to its own firing rate and the b_r,b_l,b_u, b_d in Equation 4.4 will also be more linear corresponding to input velocity.

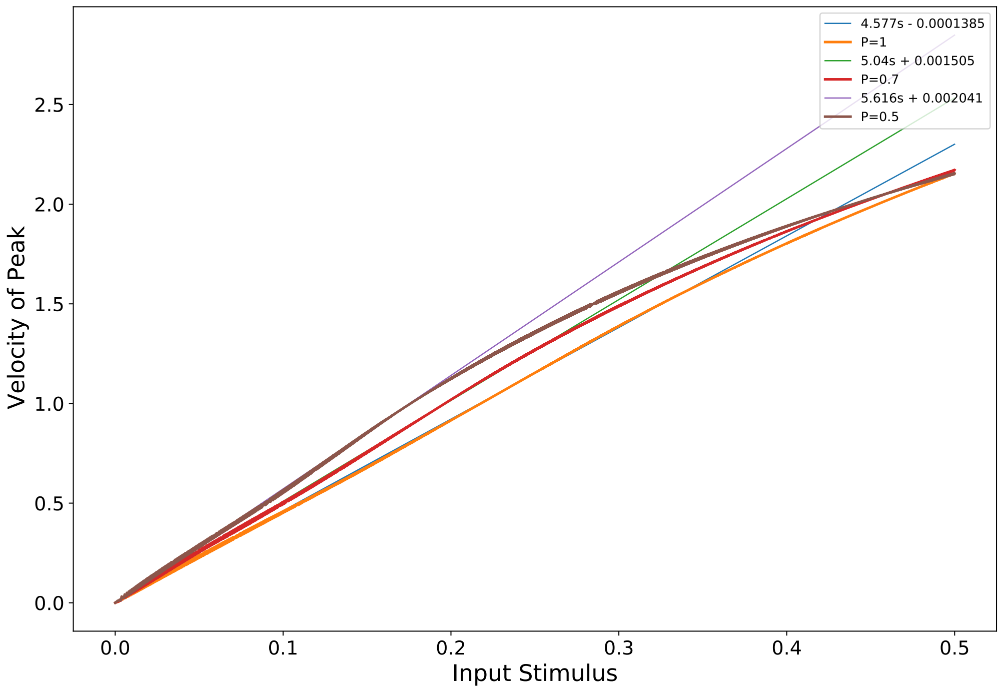

Figure 5.7.: Velocity of peak, which is decoded from Equation 4.19, with $p = 0.5$, $p = 0.7$ and $p = 1$ with other parameters in Table 5.1. The red lines are the linear fitted line between the speed and stimulus calculated by the least square method.

- Although the size of the linear space does not limit the range of input speed. But through experiments, we found that the larger the linear space, the smaller the error. Because when the linear space between the moving speed of peak in value space layer and the input stimulus is smaller, the corresponding large speed space will be compressed to a small input stimulus linear space. Then the accuracy of the information will be lost.

5.2.4. Simulation Result

In this simulation, we use Pybullet in subsection 5.1.1 as the physics engine and simulate the robot in subsection 5.1.2 to randomly walk in a square area, which is mentioned in subsection 5.1.4 Figure 5.3 for 8000 seconds, the simulation frequency is 20Hz and the update rate of our GC model is 400Hz. In the following content, we will show in detail the performance of the GC model in the simulation.

5.2.4.1. Input Velocity and Estimated Velocity

In this part, we will reflect the relationship between the movement speed of the peak in value space layer and the input velocity of our GC model. We use the speed of the robot in the environment as the input of the GC model, and estimate the speed of the robot by measuring the moving speed of the peak. Since the speed of our peaks can be set to different scales by w_v, so that there is a scale parameter between the estimated velocity and the velocity of the peak's movement:

$$v_{es}(t) = cv_{peak}(t) \tag{5.2}$$

Where $v_{es}(t)$ is the estimated velocity of the robot, $v_{peak}(t)$ is the velocity of the peak and c is a constant parameter.

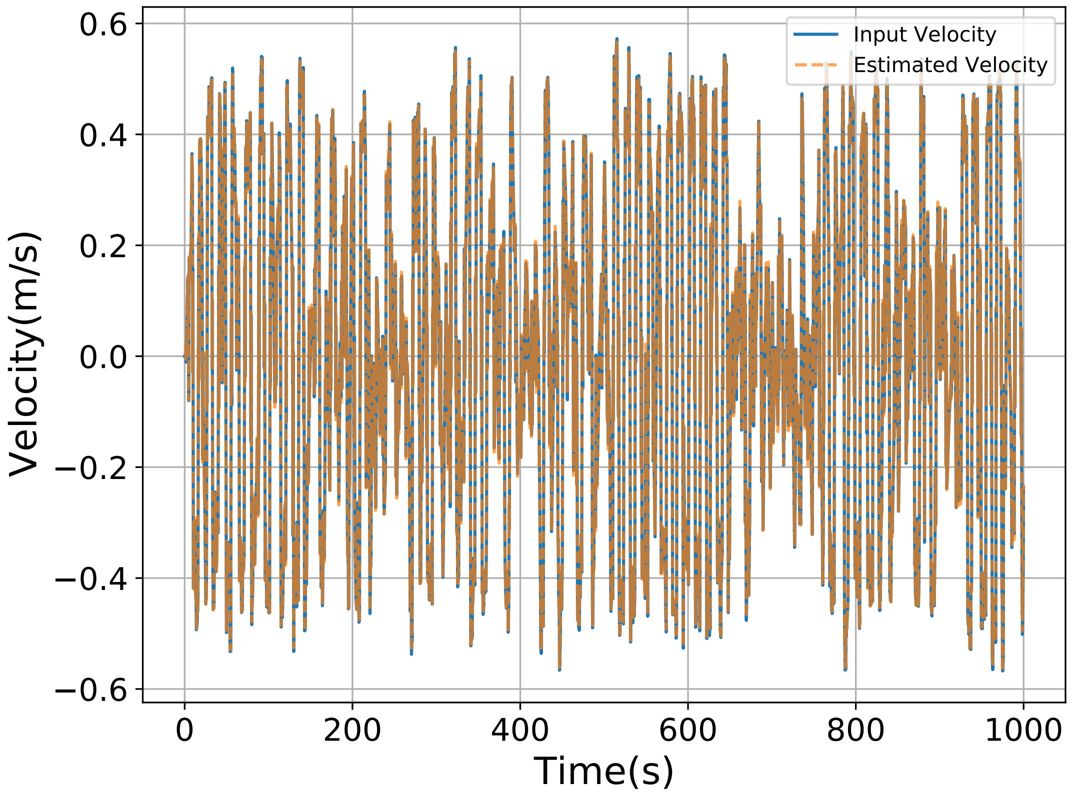

Figure 5.8.: Estimated velocity and real input velocity throughout the process. Yellow line: the estimated velocity with $c = 1.859$, blue line: real velocity of robot.

As shown in Figure 5.8 and Figure 5.9, the estimated speed during the entire simulation process is basically the same as the real input speed (robot's velocity). As shown in Figure 5.10, we measure the accuracy of the model by calculating the difference between the estimated speed and the real input speed and the distance between the real position and the estimated

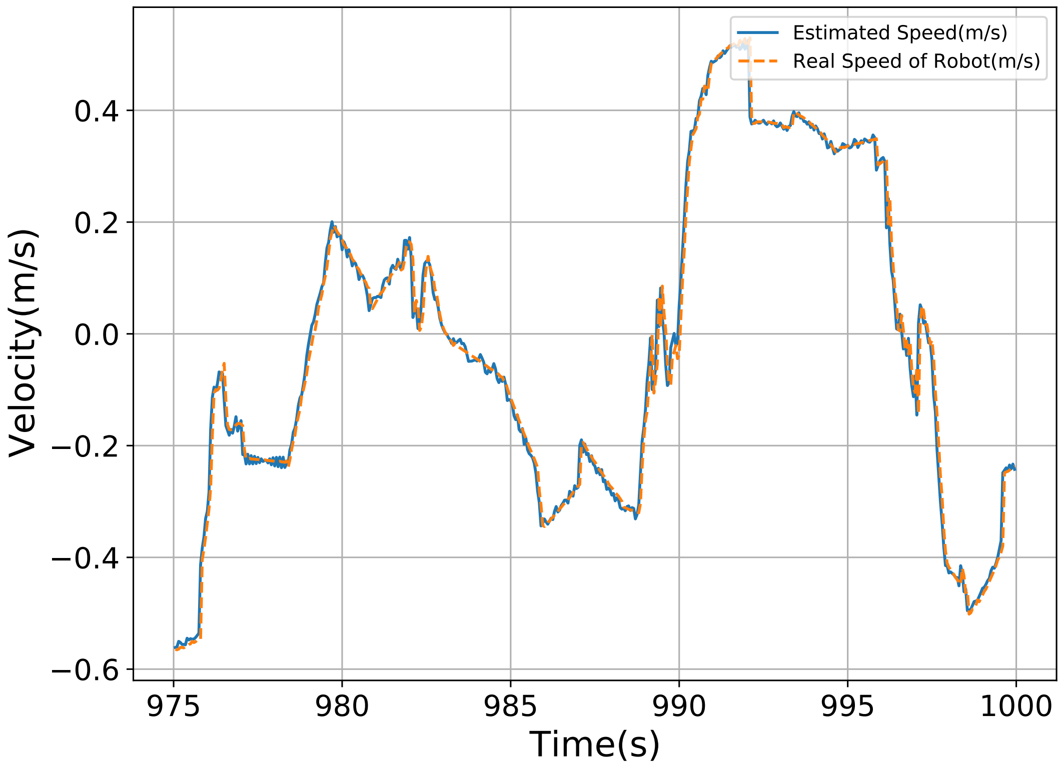

Figure 5.9.: Estimated velocity and real input velocity in last 500 seconds. Yellow line: the estimated velocity with $c = 1.859$, blue line: real velocity of robot.

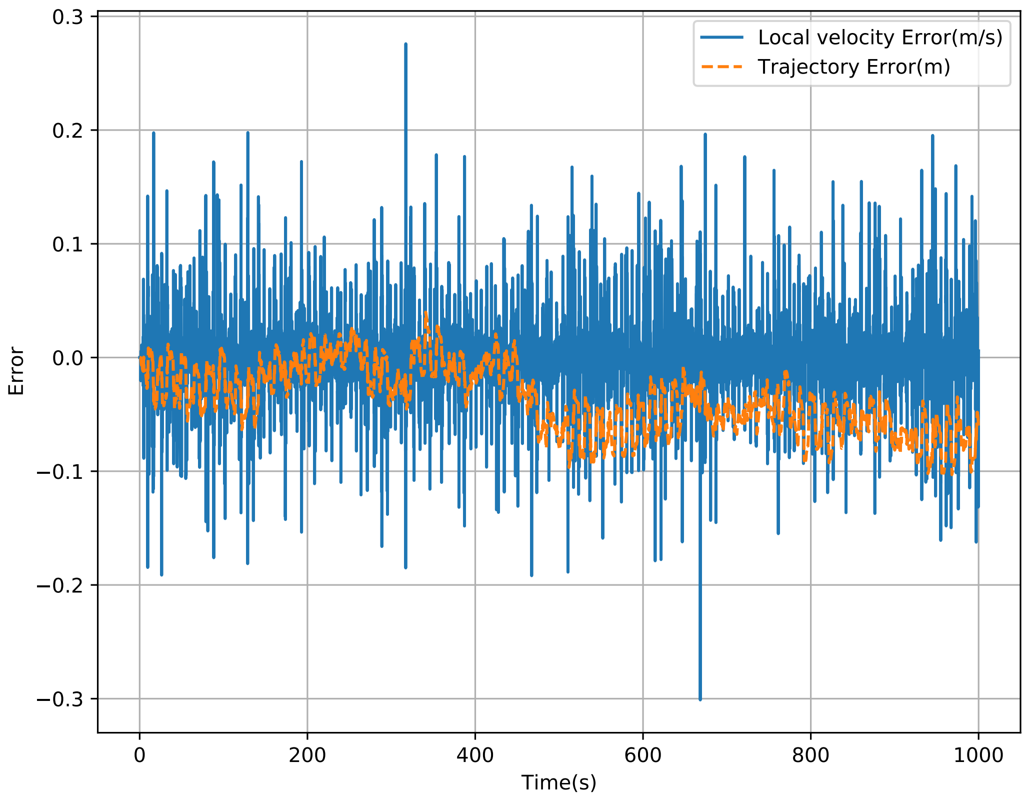

Figure 5.10.: Blue line: Error between estimated velocity and real input velocity. Yellow line:integration of error between estimated velocity and real input velocity

position of robot, which is calculated by integrating the error between the input speed and the estimated speed. We got the following conclusion from Figure 5.10:

- Although the real-time error($error(t)$) seems to be a bit large, this error will be reduced once it is integrated (the yellow line). This is because the movement of the peak is delayed relative to the input speed. At time t, we consider the velocity of robot as input, and at the next moment $t+1$ or even $t+2$, our peak will move corresponding to the input at t. Since the frequency of our neuron network is very high (400Hz), the delay has just little effect on our model.
- The integration of error (yellow line) is the distance between the actual location and the estimated location. It is very small compared to the previous model. For example, compare with [8], for the GC model in [8], the robot traveled 260 meters in 1800 seconds, resulting in an error of 15 cm. But our robot traveled 2246.44 meters in 8000 seconds, only produced an about 3.41 cm error.

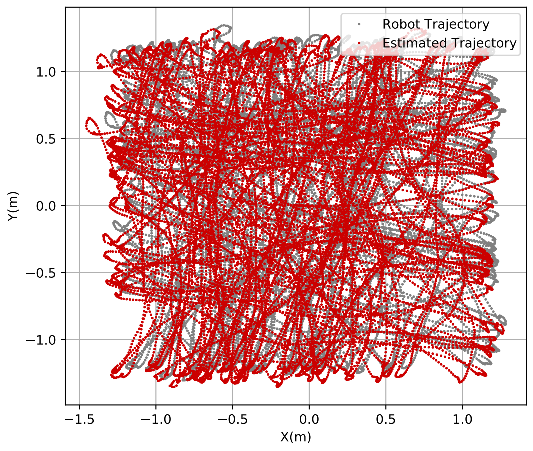

Figure 5.11.: Estimated trajectory and real trajectory throughout the simulation

From Figure 5.11 and Figure 5.12, we can see that estimated trajectory and real trajectory are very similar, which can show that the accuracy of the gird cell model is excellent.

5.2.4.2. hexagonal firing pattern

When the moving speed of peak in value space layer and the real input have a linear relationship, and it is combined with the twisted torus structure in Figure 4.3, a hexagonal firing pattern will appear. Whether a hexagonal firing pattern can be formed is an important criterion for judging the performance of the GC model. As shown in Figure 5.13, Figure 5.14 and Figure 5.15, our GC model formed hexagonal firing patterns with different spacing and orientation during the entire simulation process according to different parameters. Our

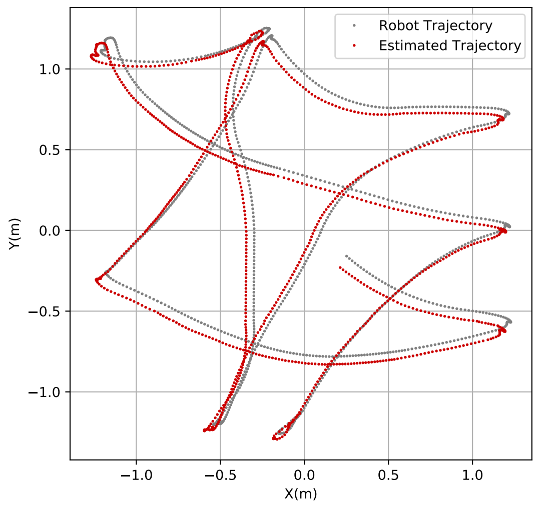

Figure 5.12.: Estimated trajectory and real trajectory in last 1000 seconds

hexagonal firing pattern shows a clear equilateral hexagon structure and firing points at vertices of hexagon is very concentrated. These show that the drift of our GC model is very small and there is a very obvious linear relationship between input and moving speed of peak in value space layer.

We can achieve the purpose of changing the shape of the hexagonal firing pattern by adjusting these parameters:

- The parameter w_v affects the spacing of the hexagonal pattern. When this parameter is larger, the input corresponding to the same speed is larger, then the moving speed of the peak will become faster, which will also cause the period of firing to become smaller. In other words, the spacing of the hexagonal firing pattern will become smaller.
- The rotation matrix R_s can change the orientation of the hexagonal firing pattern, which is shown in Figure 5.14 and Figure 5.15. In other words, the hexagonal firing pattern can be rotated.
- We can also specify the initialization of the activity of the value space layer of the GC to achieve the purpose of controlling the phase of hexagonal firing pattern. But generally in the application, we only need to randomize the initial value of the value space layer's activities.

Regardless of our hexagonal firing pattern, the linear relationship between the input velocity and peak moving velocity, stability with fixed input or without inputs, our GC model has very good performance. Its accuracy, performance and reasonable structure are fully capable of being used as part of biologically inspired robotic navigation system.

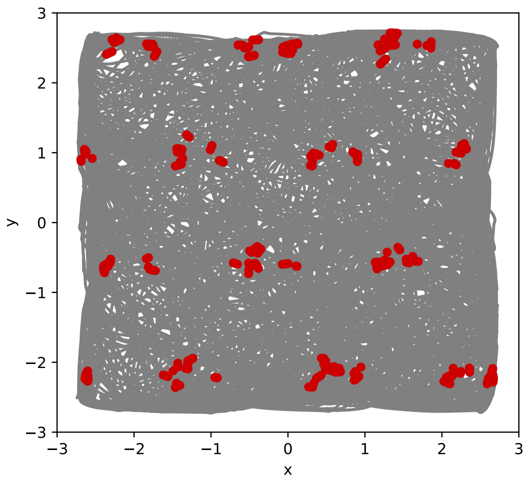

Figure 5.13.: firing map of neuron 88 with $w_v = 0.12$, $\epsilon = 0$. The gray line represents the trajectory of the agent, and the red dot indicates that the cell 88 is firing (when the firing rate of the neuron is biggest in the value space layer) at this point

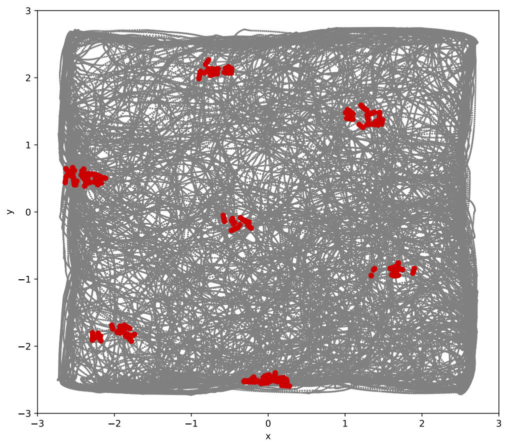

Figure 5.14.: firing map of neuron 88 with $w_v = 0.1$, $\epsilon = \pi/10$. The gray line represents the trajectory of the agent, and the red dot indicates that the cell 88 is firing (when the firing rate of the neuron is biggest in the value space layer) at this point

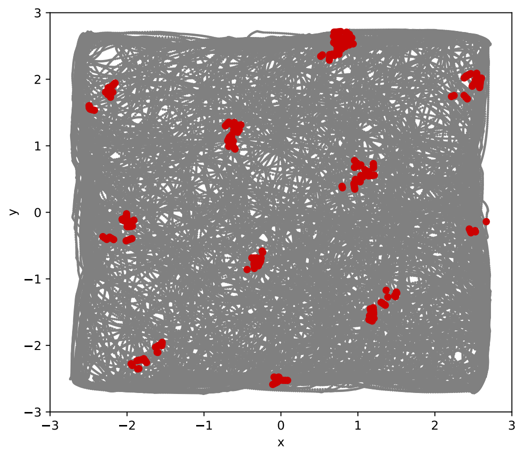

Figure 5.15.: Firing map of neuron 88 with $w_v = 0.12$, $\epsilon = \pi/10$: The gray line represents the trajectory of the agent, and the red dot indicates that the cell is firing (when the firing rate of the neuron is biggest in the value space layer) at this point.

5.3. Hebbian Learning and Place Cells

In this section we will discuss the performance and feasibility of the PC model in the case of using Hebbian learning. We use the encoded PC firing rate as the presynaptic firing rate, which is mentioned in subsection 4.5.1, and the firing rate of the GC as the postsynaptic firing rate, which is mentioned in subsection 4.5.2, to determine the synaptic weight of the connection between GC and PC through hebbian learning as shown in Equation 4.29. In the end, our learning ends after the PC can continuously predict the position successfully 1000 times.

5.3.1. Grid Cells Setting

The inputs of PC were firing rates derived from a phenomenological characterization of the activity of GC in MEC with varying phases, orientations and scales, along the trajectory followed by a agent in a walled square box ($6m \times 6m$), which is shown in Figure 5.3. The model in this article contain 1000 GC as input to 36 PC in HC. The phase, orientation, and scale of the GC were uniformly sampled as follows. The 1000 units were first divided in 10 groups of 100 units that corresponded to 10 different scales of intervertex spacing, ranging from 0.5 to 5.3m by constant increments. Each group was in turn split into 10 subgroups, each of 10 units, corresponding to 10 different orientations separated by 6° increments. The fist orientation value was sampled randomly in $0 \sim 6\circ$ range independently for each different scale. Finally, there are 10 GC left in each group. We need to move these GC with different ζ_j, which is shown in subsection 4.5.2:

$$\zeta_j = \begin{pmatrix} \zeta_j^x \\ \zeta_j^y \end{pmatrix} \tag{5.3}$$

We stipulate that the phases of 10 GC are randomly in $\zeta_j^x \in [-3, 3]$ and $\zeta_j^y \in [-3, 3]$.

5.3.2. Result

Finally, the performance of our PC model is very good. The whole area is divided into 36 areas corresponding to 36 PC as shown in Figure 4.13. Our GC uses 1000 GC with different directions, spacing and phases subsection 5.3.1, after hebbian learning, a very good performance was achieved. Each PC will only have a large firing rate in the corresponding area, while the firing rate in other places is relatively small. In Figure 5.16, the corresponding area of this cell is $x \in [-3, -2]$ and $y \in [-3, -2]$. In Figure 5.17, the corresponding area of another cell is $x \in [-1, 0]$ and $y \in [0, 1]$. These two firing maps accurately show that only in a prescribed area, the corresponding PC will have a very high firing rate. If we need a finer resolution, that is, the area corresponding to each PC is smaller, it means that, there will be more PC neurons. Then we need more GC to be able to show the specific area of each PC.

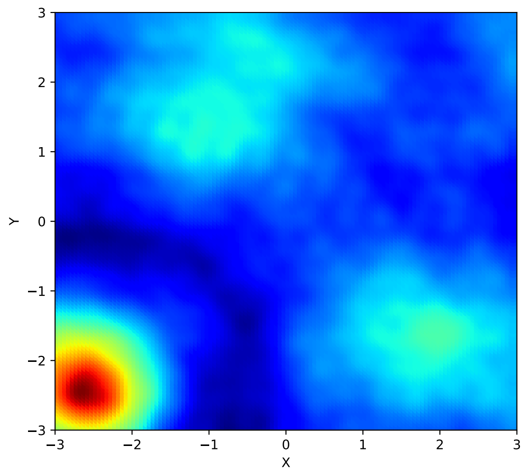

Figure 5.16.: I
Firing map of a PC neuron 1 corresponding to $x \in [-3, -2]$ and $y \in [-3, -2]$ area with 1000 different GC. Red indicates that the cell has a very large firing rate in this area. Blue means that the firing rate of the cell in this area is relatively small.

Figure 5.16 and Figure 5.17 show that our PC has very good performance. If we have a sufficiently accurate GC, we can implement the PC model through Hebbian learning.

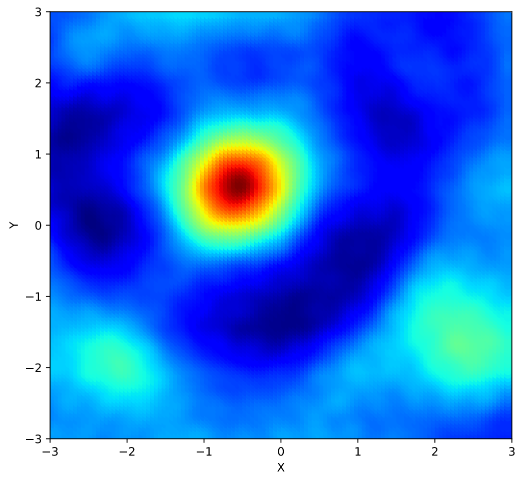

Figure 5.17.: Firing map of a PC neuron 21 corresponding to $x \in [-1, 0]$ and $y \in [0, 1]$ area with 1000 different GC. Red indicates that the cell has a very large firing rate in this area. Blue means that the firing rate of the cell in this area is relatively small.

6. Experiment

In this chapter, we will introduce how we use the information from the lidar sensor on the Four-wheeled robot to drive the robot walk randomly in different environments. Then we can get the velocity information from IMU sensor on the robot as the input of the GC model and the position of the robot in the scene in real time from a camera at a fix position in the environment. In order to reduce the velocity error of robot from IMU, we use the position information of the robot from camera to calibrate the velocity information from IMU. In this chapter, We also use the position of the robot and the activity of the GC model to generate firing maps to test the performance of our GC model. In this experiment, our GC model also shows great performance, which is driven by real data from experiment.

6.1. Experimental Scene

Our experiment conducted in an enclosed area about 2.2m by 2.2m. The initial position of the car is in the center of the area, and the initial direction is along the x direction in the environment. And in our environment, there are four markers on the ground and a camera above the ground to get the position of the robot in the area, which is shown in Figure 6.1.

6.2. Robot

Our robot consists of a Raspberry Pi, an IMU sensor, a lidar sensor, a motor controller and four motors for the wheels as shown in subsubsection 6.2.1.1, subsubsection 6.2.1.2 and Figure 6.2. Raspberry Pi runs ROS to process sensor information from lidar to control the speed of the robot's four wheels and send the speed command to the motor controller. It also subscribes and publishes sensor information, so that we can understand the specific status of the robot. The motor controller can process the command sent by the Raspberry Pi to control the power supply of the motor to achieve the purpose of controlling the speed of each wheel. Below I will separately introduce the components of the robot.

6.2.1. Sensors

In our experiment we mainly used three sensors. They are IMU, lidar, and a camera. The IMU sensor is mainly used to get the acceleration information of the robot. The lidar sensor is used to get the distance information between robot and the obstacle, so that the robot can walk randomly and avoid obstacles in different environment. The camera at a fixed position

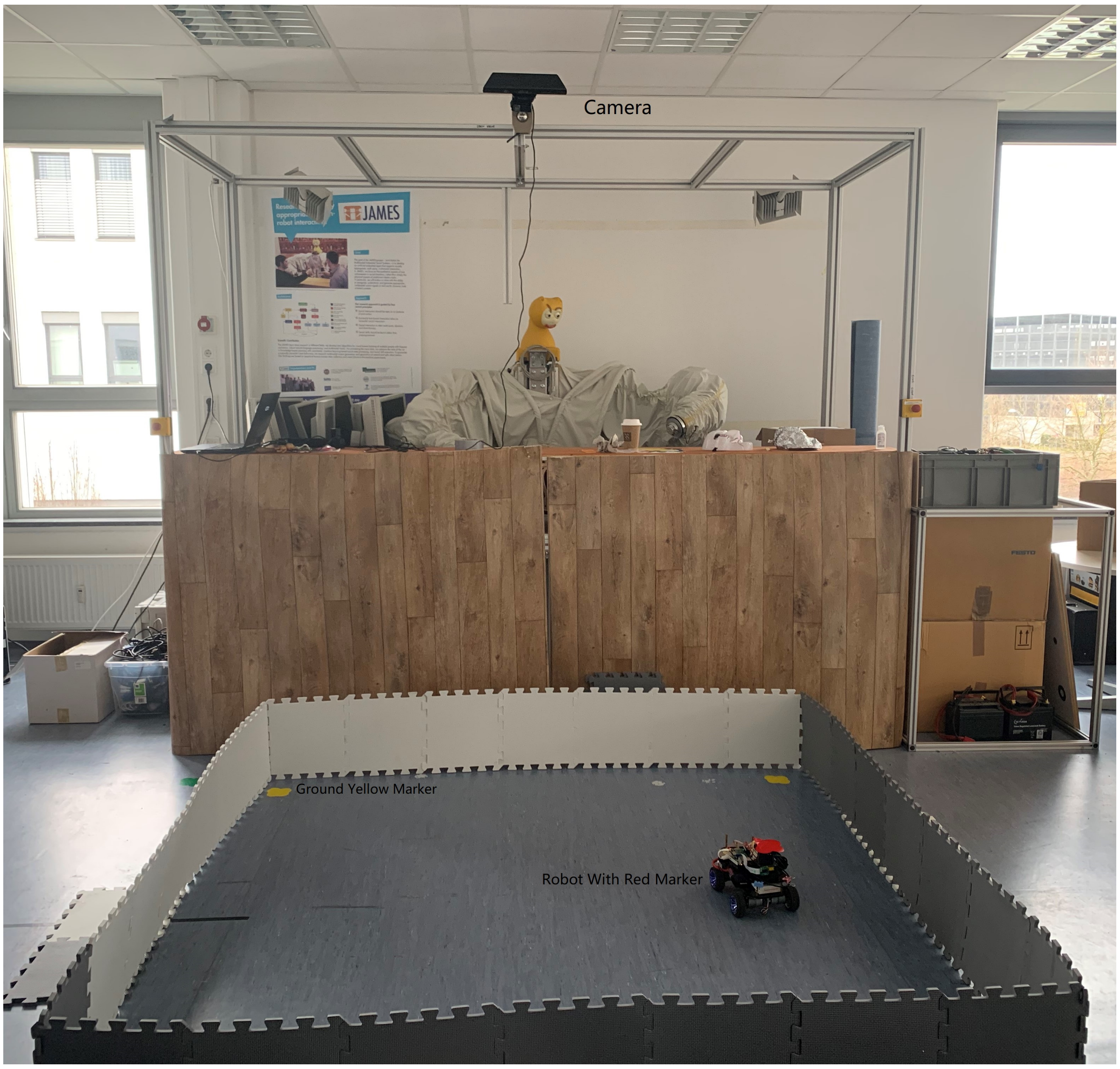

Figure 6.1.: Experimental Scene: Our experiment conducted in a closed area close to 2.2m by 2.2m. And the initial direction is on the left in the environment. There are four markers on the ground and a camera above the ground.

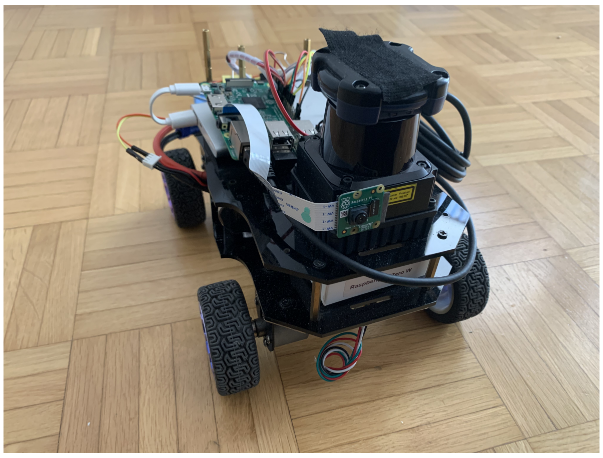

Figure 6.2.: Experimental robot: Raspberry Pi runs ROS to process sensor information to control the speed of the robot's four wheels. The robot has an IMU sensor, a lidar sensor, and a motor controller that processes the speed control commend sent by the Raspberry Pi. The four wheels of the robot are separately controlled by four motors.

in the scene is mainly complicated to monitor the position of the robot in the environment. Then I will introduce these three sensors separately in following sections.

6.2.1.1. Lidar Sensor

The lidar sensor used in our experiment is hokuyo UTM-30LX-EW scanning laser as shown in Figure 6.3. The UTM-30LX-EW is a scanning laser rangefinder that provides a sensing range to 30 meters with ethernet interface and a 25 ms scanning rate across a 270 field of view. The UTM-30LX-EW has a protective housing with a rating of IP67, making it suitable for outdoor use. The Hokuyo lasers are compact and light, making them ideal for mounting on end effectors, aerial vehicles, or small mobile platforms.

But this lidar sensor returns a large amount of data and the interval between each laser is small. Taking into account the limitations of computing power and the resulting error, we only used the data within the 180 degrees range around the robot from left to right. And we need to reduce the number of lasers. For each prefer laser shown in Figure 6.4, we calculate the average value of distance, which is in 11.25 degrees on the left and right sides of each prefer laser to get the distance between the obstacle and the robot detected by this prefer laser, which is shown in Figure 6.4.

6.2.1.2. IMU Sensor

In order to obtain the motion information of the robot, such as the linear acceleration and angular velocity of the robot, we use the IMU BNO055 sensor as shown in Figure 6.5 to get these information. This smart sensor is significantly smaller than comparable solutions.

Figure 6.3.: Lidar sensor: UTM-30LX-EW from Hokuyo

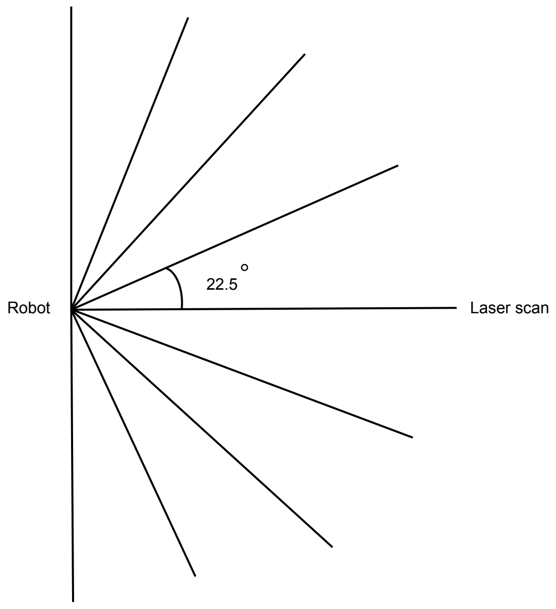

Figure 6.4.: Processed Lasers: Our lasers' range is 180 degrees from the left to the right of the car. Each laser is 22.5 degrees apart, and the detection range of each laser is from 0 to 30 meters.

By integrating sensors and sensor fusion in a single device, the BNO055 makes integration easy, avoids complex multi-vendor solutions and thus simplifies innovations, e.g. novel applications such as IoT hardware. The BNO055 is the perfect choice for AR, immersive gaming, personal health and fitness, indoor navigation and any other application requiring context awareness. It is ideally suited for demanding applications such as augmented reality, navigation, gaming, robotics, or industrial applications. In this experiment, we use linear acceleration and the measured angular velocity of the robot from the IMU to get the speed and the angle of the robot in the environment by integrating the linear acceleration and the angular velocity with the given the initial position and angle. So we get the velocity of robot as the input of GC model and positions of robot to generate firing maps.

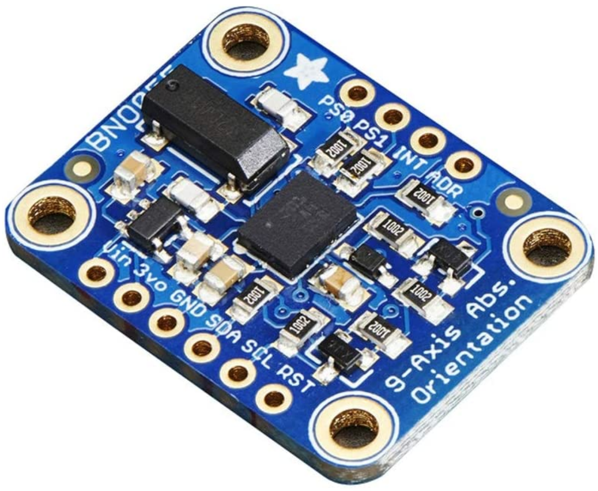

Figure 6.5.: IMU Sensor: BNO055 from bosch

6.3. Random Walk

The random walk algorithm we used is the same algorithm in subsection 5.1.4. Similar to the model in the Figure 5.4**2a**, when the left side of our robot approaches an obstacle, we give the two wheels on the left more speed, while reducing the speed of the wheels on the right. When our car approaches an obstacle on the right side, we give the right wheels more speed and reduce the speed of the left wheels. Under this setting, our robot can avoid collision with obstacles while walking in the environment. At the same time, in order to achieve better randomness, the initial speed we give to the wheels will randomly change within a certain range every 25 steps. Through experimental observation, our robot can avoid obstacles and walk randomly in different environments.

6.4. Calibration with Camera

Because the observed velocity of robot is the integration of acceleration information from IMU sensor, therefore the error of the acceleration from sensor will be accumulated in velocity and it will increase continuously. Therefore, we need additional information to calibrate the speed of the robot. In this experiment, we will use additional information from the camera, which is shown in Figure 6.6, to get the position of the robot in world coordinate system. Then we use the position of robot to calibrate the velocity of the robot. To this end, we need to make a mark on the robot body and four marks at fixed positions in the environment to get the position of the robot in this environment. In this experiment, we use different colors from the environment to identify the position of the robot and other four fixed positions in the environment. For example, we use green to mark fix positions on the ground and red to mark the robot. Then we need to define the world coordinate system and measure the positions of the four ground marks in this coordinate system. Then we use openCV [107] to identify the positions of several marker in the image. However, due to experimental conditions, it is difficult to keep the camera perpendicular to the ground, so the image captured by the camera will have perspective. In order to eliminate this perspective, we need to use openCV to calculate the perspective matrix corresponding to the image, so as to restore the position of the robot on the ground in world coordinate system by it's relative position to the fixed four markers on the ground. The final result shows that the estimated position of our algorithm has high accuracy and it can run in real time.

Figure 6.6.: Camera sensor: Xbox 360 Kinect Sensor

6.5. Result

In this experiment, we use the recorded speed of robot from IMU, and then it is calibrated by the camera information, as the input of our GC model. Combined with the observed position of the robot and activates of neurons in value space layer, we generated the firing map of the GC with different orientation and spacing as shown in Figure 6.8 Figure 6.9 and Figure 6.10. At the same time, we obtained the trajectory error of our GC model through integration of speed error as shown in Figure 6.7. As shown in Figure 6.7, Since the real data has a certain

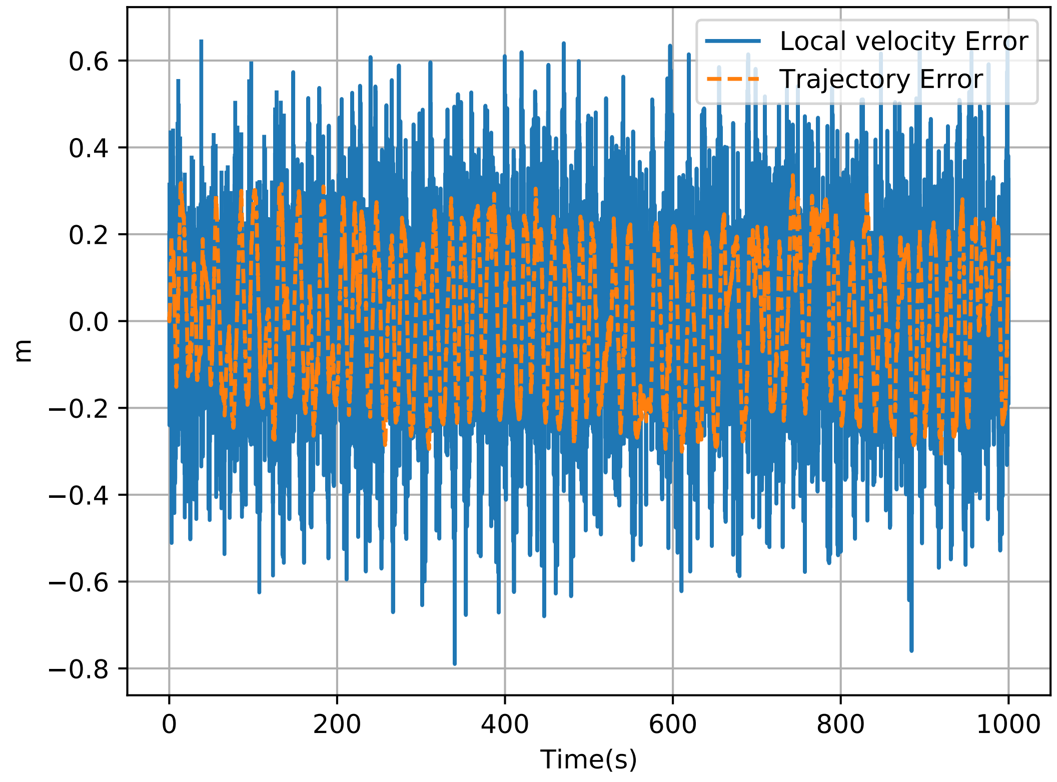

Figure 6.7.: Velocity error and position drift in 1000 seconds in experiment. Blue line: Error between estimated velocity and real input velocity($error(t) = v_{es}(t) - v(t)$). Yellow line:integration of error between estimated velocity and real input velocity. Please refer to subsubsection 5.2.4.1 for the specific method of calculating error

error and the acceleration of the robot is relatively fast, the error is relatively large compared with simulation in Figure 5.10, but it is also accurate enough. Our trajectory error fluctuates between -0.2m and 0.2m. Due to the randomness of our speed and the normalization of our model, errors will not superimpose in one direction, so that errors fluctuate within a certain range and do not divergent.

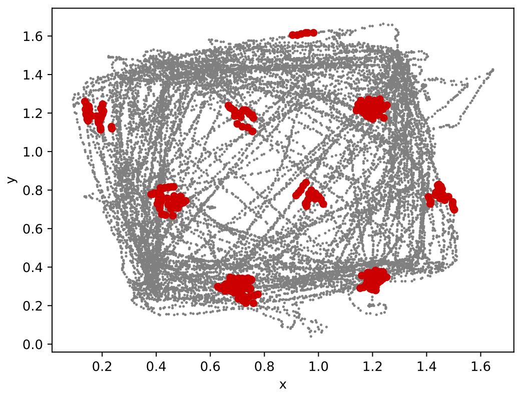

Figure 6.8.: Firing map of neuron 88 with $w_v = 0.08$, $\epsilon = 0$ in 16000 steps. The gray line represents the trajectory of the agent, and the red dot indicates that the cell 88 is firing (when the firing rate of the neuron is biggest in the value space layer) at this point

In addition, firing maps also prove that our GC model has very good performance. As shown in Figure 6.8 Figure 6.9 and Figure 6.10, the regular hexagon structure with different spacing and orientation is very obvious. (The method of adjusting spacing and direction of firing map is in subsubsection 5.2.4.2.) This is also the experimental result we hope to achieve. Therefore, whether in simulation or in experiments, our GC model shows good stability and accuracy with very small neuron network.

6.6. Supplementary Experiment

In addition, we try to use different scenarios to test our GC model. We mainly use T-maze and Y-maze, which are shown in Figure 6.12 and Figure 6.11. We used the same method as mentioned before to get the firing map in these two scenarios as shown in Figure 6.14 and Figure 6.13. As we can see from the figure Figure 6.14 and Figure 6.13, although due to the limitations of the scene, the whole hexagonal structure cannot be displayed. But the grid structure of firing map is still very obvious. This proves that our GC still also has a very good performance in complex environments.

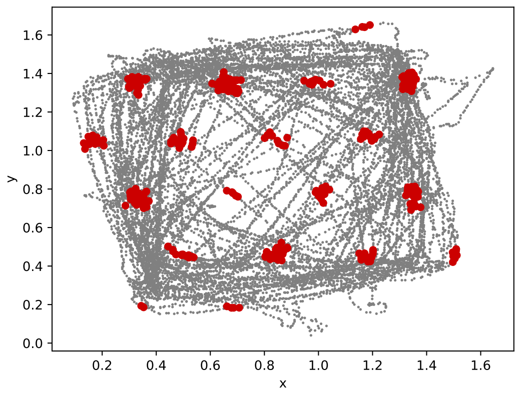

Figure 6.9.: Firing map of neuron 88 with $w_v = 0.15$, $\epsilon = 0$ in 16000 steps. The gray line represents the trajectory of the agent, and the red dot indicates that the cell 88 is firing (when the firing rate of the neuron is biggest in the value space layer) at this point

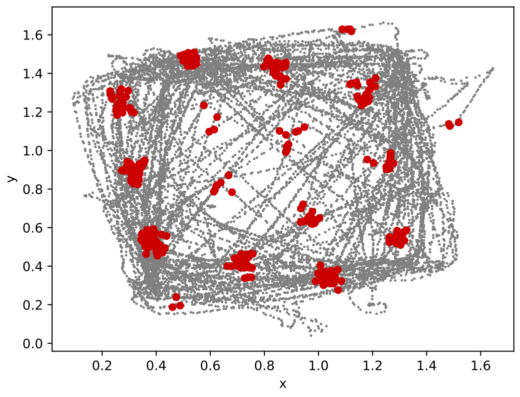

Figure 6.10.: Firing map of neuron 88 with $w_v = 0.15$, $\epsilon = \pi/10$ in 16000 steps. The gray line represents the trajectory of the agent, and the red dot indicates that the cell 88 is firing (when the firing rate of the neuron is biggest in the value space layer) at this point

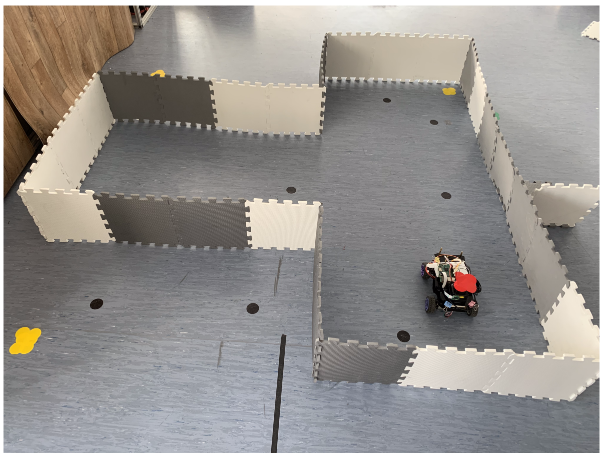

Figure 6.11.: T-maze in experiment.

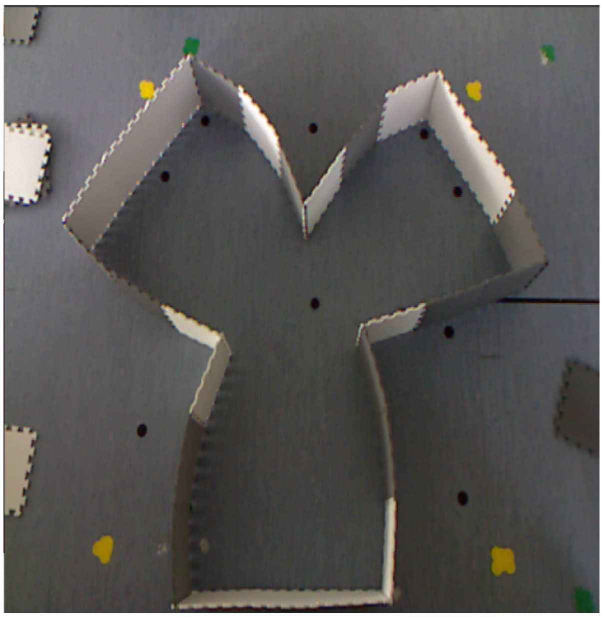

Figure 6.12.: Y-maze in experiment.

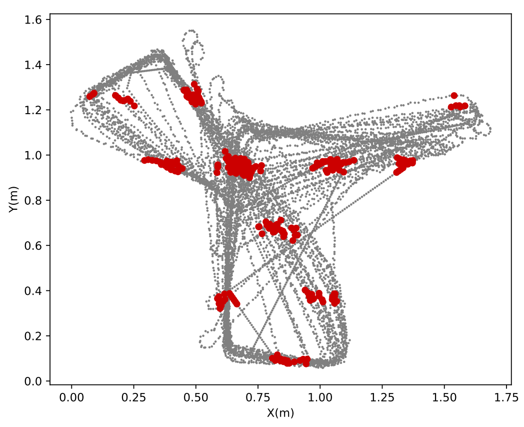

Figure 6.13.: Firing map of neuron 66 with $w_v = 0.08$ in Y-maze as shown in Figure 6.11, $\epsilon = 0$ in 16000 steps. The gray line represents the trajectory of the agent, and the red dot indicates that the cell 66 is firing (when the firing rate of the neuron is biggest in the value space layer) at this point.

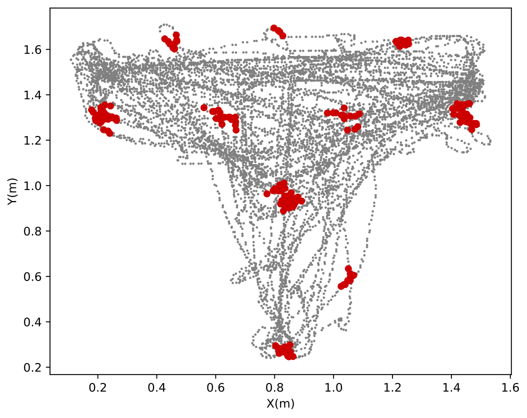

Figure 6.14.: Firing map of neuron 66 with $w_v = 0.08$ in T-maze as shown in Figure 6.12, $\epsilon = 0$ in 16000 steps. The gray line represents the trajectory of the agent, and the red dot indicates that the cell 66 is firing (when the firing rate of the neuron is biggest in the value space layer) at this point.

7. Conclusion and Future Work

7.1. Conclusion

In this article, We propose a model that can explain and restore the neural activity of GC in MEC and we have determined the PC model, as well as the connection relationship and strength between GC and PC. We have presented a model of GC based on a twisted torus topology and shift layers that generates regular triangular tessellations, as observed GC in MEC. In this model, the grids share the same orientation and spacing as observed in physiological recordings of neighboring GC in MEC. The structure of the GC model based on the previous GC research and compared with previous models [12][74][8], the performance of the model in this article is Outstanding. We showed that a simple gain w_v and rotation matrix R_s can control the grid spacing and orientation respectively, and there are different phases between neighboring GC. Thus our model provides a parsimonious explanation of how cortical circuits can give rise to GC with different spacing and orientations using a single algorithm. In the MEC, the spacing of the grid isometrically increases along the dorsoventral axis. Our model explains: this effect is due to an exponential increase of the velocity gain along this axis. We can adjust different parameters in the model to generate different shapes of hexagons. There have been many previous studies that have implemented the GC model based on attractor network. The first research to apply these methods to GC was in [3]. It has been implemented in [21] as a symmetric locally connected neural network. We have shown that this synaptic architecture that can be represented by a twisted torus Figure 4.3 generates regular triangular tessellating patterns. The advantages of our model is that it allows implementing a representation of space covering large environments using a relatively small population of cells. Moreover, because of this particular circular synaptic connectivity, all network cells have regular triangular tessellating subfields. It is also important: Our mathematical model is as close to the structure of the brain as possible. Although we can see that the model in [12] has very good results, it directly changes the synaptic weights of the connections between neurons. We know that synaptic weights between neurons in the brain are difficult to change very quickly and accurately with an input. Our model uses the activity of the shift layer to move the peak in the value space layer.

Our model of GC can be used as the proprioceptive odometer of a robust, modulatory and biologically based navigational system combining idiothetic and allothetic information. Because we are using a bionic system instead of a traditional second-generation neuron system. Therefore, the visual recognition rate is not very high. For instance, realistic models of PC by HC, based on visual inputs are not able to distinguish between two visually similar places. When using the model in this article, no visual information is needed, only some speed information is needed to assist in judging which similar place it appears in.

At the same time, based on the input of GC, we also completed the construction of the PC model. However, the implementation of PC requires a large number of GCs with different orientations,phase and spacing as a basis. Although the neurons of our GC is less enough, it is still difficult to run thousands of GCs at the same time. After we use an ideal GC encoder, and based on Hebbian Learning, we can determine the weight of the connection between the GC and the PC. Using these weight connections, our PC also has very good performance.

In short, our model is not only closer to the real biological model, but also improves the accuracy of the GC compared to some previous models [12][74][8]. and can be used as an important component a of brain-inspired navigation system.

7.2. Future Work

7.2.1. Calibration Mechanisms for Grid cells

Now our GC and PC models are all running in the dark. That is to say, our model only uses information from itself, and does not obtain information from the environment, that is, its own linear velocity and its own angular velocity as input to the model.This will inevitably lead to inaccuracy and drift of the peak position in GC and PC. First of all, in the real world, both animals and robots have errors in their perception of their own speed when they are moving. Once this error continues to add up, the accuracy of our GC will drop significantly. At the same time, the accuracy of the PC will be greatly reduced. Second, there will be certain errors in our neuron model. These errors will also be superimposed under certain special circumstances, which will cause the peak position to drift. So we need to obtain information from the environment to correct and eliminate these errors. From [108] we know that: error accumulates relative to time and distance traveled since the animal last encountered a boundary. This error reflects coherent drift in the grid pattern. Further, interactions with boundaries yield direction-dependent error correction, suggesting that border cells serve as a neural substrate for error correction. error accumulates relative to time and distance traveled since the animal last encountered a boundary. This error reflects coherent drift in the grid pattern. Further, interactions with boundaries yield direction-dependent error correction, suggesting that border cells serve as a neural substrate for error correction from [108]. This involves another kind of border cells that exist in the brain. Which is an entorhinal cell type that fires when an animal is close to the borders of the proximal environment. The orientation-specific edge-apposing activity of these "border cells" is maintained when the environment is stretched and during testing in enclosures of different size and shape in different rooms. Border cells are relatively sparse, making up less than $\frac{1}{10}$ of the local cell population, but can be found in all layers of the medial EC as well as the adjacent parasubiculum, often intermingled with head-direction cells and GC. Border cells may be instrumental in planning trajectories and anchoring grid fields and place fields to a geometric reference frame and has the function of calibration [5]. From Figure 7.1 in [5] we know that: The border cell will fire when the mouse is on the border in a specific direction. And we also found that they are usually mixed in GC and PCs. It also proves that border cells are part of GC and PCs.

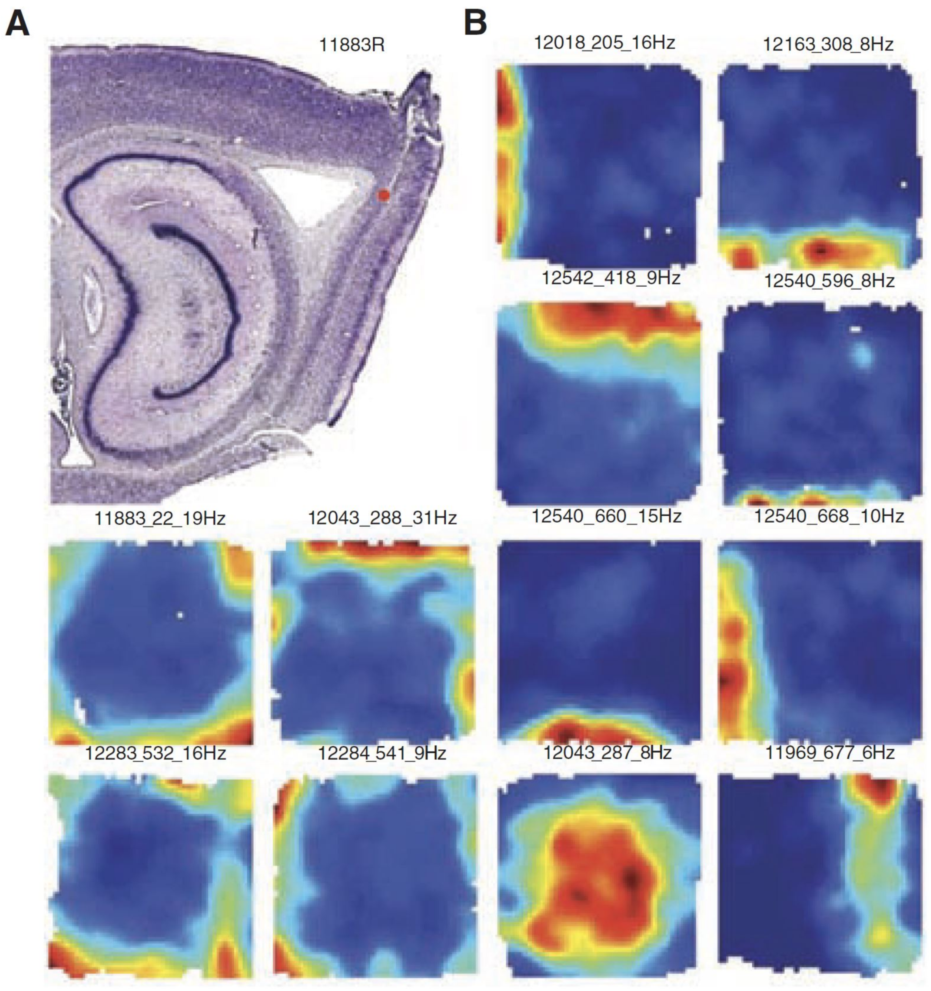

Figure 7.1.: Examples of border cells in the MEC and adjacent parasubiculum. (A) Sagittal Nissl-stained section showing a representative recording location in the MEC (red dot, recording location; rat number and hemisphere (R, right) are indicated(B) Color-coded rate maps for 12 border cells. Red is maximum, dark blue is zero. Pixels not covered are white. Animal numbers (five digits), cell numbers (two or three digits), and peak firing rates are indicated above each panel. Cells 287 and 677 did not pass the criterion for border cells because the fields were located at some distance from the wall; the number of such cells was fewer than 10. [5]

And this is also very consistent with the experimental situation. According to [109], In reorientation and navigation, rats are not sensitive to details such as the color and material of obstacles. They are only sensitive to the shape of their environment. Including young babies will show the same situation.

7.2.2. A Simpler Place Cells Model Implementation

According to most of the PC research that the input of the PC we use is an ideal GC formula like Equation 4.31. That is because it is difficult to simultaneously simulate a sufficient number of GC on existing computers. In addition to changing existing models and algorithms. We should also put forward certain requirements for hardware like in subsubsection 3.1.3.3. Continuously promote the technological update of hardware suitable for CANN model calculation.

A. Implementation Details

Our algorithm is implemented in Python (version 3.6) and tested on Ubuntu 18.04. We use a numpy array to store the firing rate of each neuron. And update their firing rate through the different weight between them. But we have a large number of neurons (1800), if we only use cpu for calculation, then our simulation speed will be very slow (1Hz). Fortunately, many of our calculations are parallel calculations, so we can also use GPUs to accelerate our simulations. we use pycuda as a bridge connecting python and cuda.

A.1. Pycuda

CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model launched by graphics card manufacturer NVIDIA in 2007. It uses the power of the GPU to achieve a significant increase in computing performance. CUDA is a general-purpose parallel computing architecture launched by NVIDIA, which enables GPUs to solve complex computing problems, so that the underlying hardware can be controlled by programs to perform calculations. It includes the CUDA instruction set architecture (ISA) and the parallel computing engine inside the GPU. Developers can use languages such as C/C++ to write programs for the CUDA architecture. In our program, we use C++ as the core function. CUDA provides a host-device programming model and a large number of interface functions and scientific computing libraries to achieve parallelism by executing a large number of threads at the same time.

The structure of the graphics card determines that it has a very powerful parallel computing capability. Because cpu has only a few threads, although they have powerful computing power, they are not suitable for parallel computing. Generally speaking, the amount of calculation in each parallel calculation is very small, so the cpu will frequently retrieve data from the memory, which will take a lot of time. But GPU has many cores. Although its core computing power and cpu are not of the same magnitude. But its computing power can still solve the problems in parallel computing. In this way, multiple cores can calculate at the same time, reducing the time to read data from the memory. Pycuda acts as a bridge between python and cuda programs, transferring the information in the numpy array to the gpu for calculation. But we still use C++ to complete the calculation functions and thread allocation in our GPU.

A.1.1. Skcuda

Because we have turned each layer into a numpy array, our update of the firing rate of each neuron involves the calculation of the matrix. In order to efficiently use the GPU, we use skcuda to perform matrix calculations on the GPU. It can better allocate the number of CPU threads and blocks to achieve efficient calculations.

A.2. Modular

In order to be more compatible with future work, our program has adopted a modular design. It is mainly divided into the following modules:

- Module for grid cell network generation In this module we will build the entire network, including initializing the fring rate of each cell and the strength of the synaptic weight between them.
- Module for update of firing rate: In this module we use pycuda to update the firing rate of each neuron. Experiments show that if we use gpu, the speed is 40 times faster than when we only use cpu.
- Module for PC model: In this module, we can use Hebbian Learning to train synaptic weights between GC and PC. And in this module, we can implement the PC model through the trained weight.

List of Figures

List of Tables

Bibliography

[1] R. G. Morris, P. Garrud, J. a. Rawlins, and J. O'Keefe. "Place navigation impaired in rats with hippocampal lesions". In: *Nature* 297.5868 (1982), pp. 681–683.

[2] J. O'Keefe and J. Dostrovsky. "The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat." In: *Brain research* (1971).

[3] T. Hafting, M. Fyhn, S. Molden, M.-B. Moser, and E. I. Moser. "Microstructure of a spatial map in the entorhinal cortex". In: *Nature* 436.7052 (2005), pp. 801–806.

[4] J. B. Ranck. "Head direction cells in the deep cell layer of dorsolateral pre-subiculum in freely moving rats". In: *Electrical activity of the archicortex* (1985).

[5] T. Solstad, C. N. Boccara, E. Kropff, M.-B. Moser, and E. I. Moser. "Representation of geometric borders in the entorhinal cortex". In: *Science* 322.5909 (2008), pp. 1865–1868.

[6] J. Ye, M. P. Witter, M.-B. Moser, and E. I. Moser. "Entorhinal fast-spiking speed cells project to the hippocampus". In: *Proceedings of the National Academy of Sciences* 115.7 (2018), E1627–E1636.

[7] T. Solstad, E. I. Moser, and G. T. Einevoll. "From grid cells to place cells: a mathematical model". In: *Hippocampus* 16.12 (2006), pp. 1026–1031.

[8] Y. Burak and I. R. Fiete. "Accurate path integration in continuous attractor network models of grid cells". In: *PLoS Comput Biol* 5.2 (2009), e1000291.

[9] A. Samsonovich and B. L. McNaughton. "Path integration and cognitive mapping in a continuous attractor neural network model". In: *Journal of Neuroscience* 17.15 (1997), pp. 5900–5920.

[10] W. Gerstner and W. M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[11] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit". In: *Nature* 405.6789 (2000), pp. 947–951.

[12] A. Guanella, D. Kiper, and P. Verschure. "A model of grid cells based on a twisted torus topology". In: *International journal of neural systems* 17.04 (2007), pp. 231–240.

[13] T. H. Brown, E. W. Kairiss, and C. L. Keenan. "HEBBIAN SYNAPSES: Biophysical Mechanisms and Algorithms". In: *Annu. Rev. Neurosci* 13 (1990), pp. 475–511.

[14] W. Maass. "Networks of spiking neurons: the third generation of neural network models". In: *Neural networks* 10.9 (1997), pp. 1659–1671.

[15] J. Misra and I. Saha. “Artificial neural networks in hardware: A survey of two decades of progress”. In: *Neurocomputing* 74.1-3 (2010), pp. 239–255.

[16] E. C. Tolman. “Cognitive maps in rats and men.” In: *Psychological review* 55.4 (1948), p. 189.

[17] J. O’Keefe and D. Conway. “Hippocampal place units in the freely moving rat: why they fire where they fire”. In: *Experimental brain research* 31.4 (1978), pp. 573–590.

[18] J. O’keefe and L. Nadel. *The hippocampus as a cognitive map*. Oxford: Clarendon Press, 1978.

[19] J. O’Keefe and A. 1. Speakman. “Single unit activity in the rat hippocampus during a spatial memory task”. In: *Experimental brain research* 68.1 (1987), pp. 1–27.

[20] A. S. Etienne and K. J. Jeffery. “Path integration in mammals”. In: *Hippocampus* 14.2 (2004), pp. 180–192.

[21] M. C. Fuhs and D. S. Touretzky. “A spin glass model of path integration in rat medial entorhinal cortex”. In: *Journal of Neuroscience* 26.16 (2006), pp. 4266–4276.

[22] K. J. Jeffery and J. M. O’Keefe. “Learned interaction of visual and idiothetic cues in the control of place field orientation”. In: *Experimental Brain Research* 127.2 (1999), pp. 151–161.

[23] B. L. McNaughton, S. Mizumori, C. A. Barnes, B. Leonard, M. Marquis, and E. Green. “Cortical representation of motion during unrestrained spatial navigation in the rat”. In: *Cerebral Cortex* 4.1 (1994), pp. 27–39.

[24] H. Mittelstaedt and M.-L. Mittelstaedt. “Homing by path integration”. In: *Avian navigation*. Springer, 1982, pp. 290–297.

[25] B. Poucet. “Spatial cognitive maps in animals: new hypotheses on their structure and neural mechanisms.” In: *Psychological review* 100.2 (1993), p. 163.

[26] M. Fyhn, S. Molden, M. P. Witter, E. I. Moser, and M.-B. Moser. “Spatial representation in the entorhinal cortex”. In: *Science* 305.5688 (2004), pp. 1258–1264.

[27] E. I. Moser, E. Kropff, and M.-B. Moser. “Place cells, grid cells, and the brain’s spatial representation system”. In: *Annu. Rev. Neurosci.* 31 (2008), pp. 69–89.

[28] J. Ranck Jr. “Head direction cells in the deep layer of dorsal presubiculum in freely moving rats”. In: *Society of Neuroscience Abstract*. Vol. 10. 1984, p. 599.

[29] D. A. Henze, Z. Borhegyi, J. Csicsvari, A. Mamiya, K. D. Harris, and G. Buzsaki. “Intracellular features predicted by extracellular recordings in the hippocampus in vivo”. In: *Journal of neurophysiology* 84.1 (2000), pp. 390–400.

[30] J. O’Keefe. “Place units in the hippocampus of the freely moving rat”. In: *Experimental neurology* 51.1 (1976), pp. 78–109.

[31] M. A. Wilson and B. L. McNaughton. “Dynamics of the hippocampal ensemble code for space”. In: *Science* 261.5124 (1993), pp. 1055–1058.

[32] D. M. Smith and S. J. Mizumori. “Hippocampal place cells, context, and episodic memory”. In: *Hippocampus* 16.9 (2006), pp. 716–729.

[33] L. Nadel. “The hippocampus and space revisited”. In: *Hippocampus* 1.3 (1991), pp. 221–229.

[34] E. T. Rolls. “Spatial view cells and the representation of place in the primate hippocampus”. In: *Hippocampus* 9.4 (1999), pp. 467–480.

[35] M.-B. Moser, D. C. Rowland, and E. I. Moser. “Place cells, grid cells, and memory”. In: *Cold Spring Harbor perspectives in biology* 7.2 (2015), a021808.

[36] J. S. Taube, R. U. Muller, and J. B. Ranck. “Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis”. In: *Journal of Neuroscience* 10.2 (1990), pp. 420–435.

[37] E. Markus, B. McNaughton, C. Barnes, J. Green, and J. Meltzer. “Head direction cells in the dorsal presubiculum integrate both visual and angular velocity information”. In: *Soc. Neurosci. Abstr*. Vol. 16. 1990, p. 441.

[38] S. Mizumori and J. D. Williams. “Directionally selective mnemonic properties of neurons in the lateral dorsal nucleus of the thalamus of rats”. In: *Journal of Neuroscience* 13.9 (1993), pp. 4015–4028.

[39] J. S. Taube. “The head direction signal: origins and sensory-motor integration”. In: *Annu. Rev. Neurosci.* 30 (2007), pp. 181–207.

[40] B. McNaughton, E. Markus, M. Wilson, and J. Knierim. “Familiar landmarks can correct for cumulative error in the inertially based deadreckoning system”. In: *Society for Neuroscience Abstracts*. Vol. 19. 1993, p. 795.

[41] J. Goodridge, K. Worboys, P. Dudchenko, and J. Taube. “The response of head direction cells to non-visual cues”. In: *Soc Neurosci Abstr*. Vol. 21. 1995, p. 945.

[42] J. S. Taube and H. L. Burton. “Head direction cell activity monitored in a novel environment and during a cue conflict situation”. In: *Journal of Neurophysiology* 74.5 (1995), pp. 1953–1971.

[43] B. McNaughton, L. Chen, and E. Markus. ““Dead reckoning,” landmark learning, and the sense of direction: a neurophysiological and computational hypothesis”. In: *Journal of cognitive neuroscience* 3.2 (1991), pp. 190–202.

[44] C. A. Barnes, B. L. McNaughton, S. J. Mizumori, B. W. Leonard, and L.-H. Lin. “Chapter Comparison of spatial and temporal characteristics of neuronal activity in sequential stages of hippocampal processing”. In: *Progress in brain research*. Vol. 83. Elsevier, 1990, pp. 287–300.

[45] G. J. Quirk, R. U. Muller, J. L. Kubie, and J. B. Ranck. “The positional firing properties of medial entorhinal neurons: description and comparison with hippocampal place cells”. In: *Journal of Neuroscience* 12.5 (1992), pp. 1945–1963.

[46] B. L. Mcnaughton, C. A. Barnes, J. Meltzer, and R. Sutherland. "Hippocampal granule cells are necessary for normal spatial learning but not for spatially-selective pyramidal cell discharge". In: *Experimental brain research* 76.3 (1989), pp. 485–496.

[47] V. H. Brun, M. K. Otnæss, S. Molden, H.-A. Steffenach, M. P. Witter, M.-B. Moser, and E. I. Moser. "Place cells and place recognition maintained by direct entorhinal-hippocampal circuitry". In: *Science* 296.5576 (2002), pp. 2243–2246.

[48] F. Sargolini, M. Fyhn, T. Hafting, B. L. McNaughton, M. P. Witter, M.-B. Moser, and E. I. Moser. "Conjunctive representation of position, direction, and velocity in entorhinal cortex". In: *Science* 312.5774 (2006), pp. 758–762.

[49] S.-i. Amari. "Dynamics of pattern formation in lateral-inhibition type neural fields". In: *Biological cybernetics* 27.2 (1977), pp. 77–87.

[50] C. Eliasmith. "Attractor network". In: *Scholarpedia* 2.10 (2007), p. 1380.

[51] P. C. Bressloff. "Spatiotemporal dynamics of continuum neural fields". In: *Journal of Physics A: Mathematical and Theoretical* 45.3 (2011), p. 033001.

[52] S. Wu, K. M. Wong, C. A. Fung, Y. Mi, and W. Zhang. "Continuous attractor neural networks: candidate of a canonical model for neural information representation". In: *F1000Research* 5 (2016).

[53] A. P. Georgopoulos, M. Taira, and A. Lukashin. "Cognitive neurophysiology of the motor cortex". In: *Science* 260.5104 (1993), pp. 47–52.

[54] R. Ben-Yishai, R. L. Bar-Or, and H. Sompolinsky. "Theory of orientation tuning in visual cortex." In: *Proceedings of the National Academy of Sciences* 92.9 (1995), pp. 3844–3848.

[55] K. Zhang. "Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory". In: *Journal of Neuroscience* 16.6 (1996), pp. 2112–2126.

[56] A. Ponce-Alvarez, A. Thiele, T. D. Albright, G. R. Stoner, and G. Deco. "Stimulus-dependent variability and noise correlations in cortical MT neurons". In: *Proceedings of the National Academy of Sciences* 110.32 (2013), pp. 13162–13167.

[57] K. Wimmer, D. Q. Nykamp, C. Constantinidis, and A. Compte. "Bump attractor dynamics in prefrontal cortex explains behavioral precision in spatial working memory". In: *Nature neuroscience* 17.3 (2014), pp. 431–439.

[58] S. S. Kim, H. Rouault, S. Druckmann, and V. Jayaraman. "Ring attractor dynamics in the Drosophila central brain". In: *Science* 356.6340 (2017), pp. 849–853.

[59] D. J. Amit and D. J. Amit. *Modeling brain function: The world of attractor neural networks*. Cambridge university press, 1992.

[60] J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[61] S. Wu and S.-I. Amari. “Computing with continuous attractors: stability and online aspects”. In: *Neural computation* 17.10 (2005), pp. 2215–2239.

[62] S. Wu, K. Hamaguchi, and S.-i. Amari. “Dynamics and computation of continuous attractors”. In: *Neural computation* 20.4 (2008), pp. 994–1025.

[63] S. Wu, S.-i. Amari, and H. Nakahara. “Population coding and decoding in a neural field: a computational study”. In: *Neural Computation* 14.5 (2002), pp. 999–1026.

[64] S. Deneve, P. E. Latham, and A. Pouget. “Reading population codes: a neural implementation of ideal observers”. In: *Nature neuroscience* 2.8 (1999), pp. 740–745.

[65] S. Wu, H. Nakahara, N. Murata, and S.-i. Amari. “Population decoding based on an unfaithful model”. In: *Advances in neural information processing systems*. 2000, pp. 192–198.

[66] S. Wu, H. Nakahara, and S.-I. Amari. “Population coding with correlation and an unfaithful model”. In: *Neural computation* 13.4 (2001), pp. 775–797.

[67] W.-H. Zhang and S. Wu. “Reciprocally coupled local estimators implement Bayesian information integration distributively”. In: *Advances in neural information processing systems* 26 (2013), pp. 19–27.

[68] W.-H. Zhang, H. Wang, K. Wong, and S. Wu. ““Congruent” and “opposite” neurons: sisters for multisensory integration and segregation”. In: *Advances in neural information processing systems* 29 (2016), pp. 3180–3188.

[69] W.-H. Zhang, H. Wang, A. Chen, Y. Gu, T. S. Lee, K. M. Wong, and S. Wu. “Complementary congruent and opposite neurons achieve concurrent multisensory integration and segregation”. In: *Elife* 8 (2019), e43753.

[70] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, et al. “Towards artificial general intelligence with hybrid Tianjic chip architecture”. In: *Nature* 572.7767 (2019), pp. 106–111.

[71] A. Barrera, G. Tejera, M. Llofriu, and A. Weitzenfeld. “Learning spatial localization: from rat studies to computational models of the hippocampus”. In: *Spatial Cognition & Computation* 15.1 (2015), pp. 27–59.

[72] A. Barrera and A. Weitzenfeld. “Biologically-inspired robot spatial cognition based on rat neurophysiological studies”. In: *Autonomous Robots* 25.1-2 (2008), pp. 147–169.

[73] M. A. Brown and P. E. Sharp. “Simulation of spatial learning in the Morris water maze by a neural network model of the hippocampal formation and nucleus accumbens”. In: *Hippocampus* 5.3 (1995), pp. 171–188.

[74] N. Burgess, M. Recce, and J. O’Keefe. “A model of hippocampal function”. In: *Neural networks* 7.6-7 (1994), pp. 1065–1081.

[75] K. Caluwaerts, M. Staffa, S. N’Guyen, C. Grand, L. Dollé, A. Favre-Félix, B. Girard, and M. Khamassi. “A biologically inspired meta-control navigation system for the psikharpax rat robot”. In: *Bioinspiration & biomimetics* 7.2 (2012), p. 025009.

[76] L. Dollé, D. Sheynikhovich, B. Girard, R. Chavarriaga, and A. Guillot. "Path planning versus cue responding: a bio-inspired model of switching between navigation strategies". In: *Biological cybernetics* 103.4 (2010), pp. 299–317.

[77] D. Filliat and J.-A. Meyer. "Global localization and topological map learning for robot navigation". In: 2002.

[78] P. Gaussier, A. Revel, J.-P. Banquet, and V. Babeau. "From view cells and place cells to cognitive map learning: processing stages of the hippocampal system". In: *Biological cybernetics* 86.1 (2002), pp. 15–28.

[79] M. Milford and G. Wyeth. "Spatial mapping and map exploitation: a bio-inspired engineering perspective". In: *International Conference on Spatial Information Theory*. Springer. 2007, pp. 203–221.

[80] M. Milford and G. Wyeth. "Persistent navigation and mapping using a biologically inspired SLAM system". In: *The International Journal of Robotics Research* 29.9 (2010), pp. 1131–1153.

[81] D. O. Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

[82] A. Arleo, F. Smeraldi, and W. Gerstner. "Cognitive navigation based on nonuniform Gabor space sampling, unsupervised growing networks, and reinforcement learning". In: *IEEE transactions on neural networks* 15.3 (2004), pp. 639–652.

[83] W. Skaggs, J. Knierim, H. Kudrimoti, and B. McNaughton. "A model of the neural basis of the rat's sense of direction". In: *Advances in neural information processing systems* 7 (1994), pp. 173–180.

[84] G. Tang, A. Shah, and K. P. Michmizos. "Spiking neural network on neuromorphic hardware for energy-efficient unidimensional SLAM". In: *arXiv preprint arXiv:1903.02504* (2019).

[85] D. C. Rowland, Y. Roudi, M.-B. Moser, and E. I. Moser. "Ten years of grid cells". In: *Annual review of neuroscience* 39 (2016), pp. 19–40.

[86] W. E. Skaggs, J. J. Knierim, H. S. Kudrimoti, and B. L. McNaughton. "A model of the neural basis of the rat's sense of direction". In: *Advances in neural information processing systems*. 1995, pp. 173–180.

[87] N. Burgess, C. Barry, and J. O'keefe. "An oscillatory interference model of grid cell firing". In: *Hippocampus* 17.9 (2007), pp. 801–812.

[88] L. M. Giocomo, M.-B. Moser, and E. I. Moser. "Computational models of grid cells". In: *Neuron* 71.4 (2011), pp. 589–603.

[89] F. Savelli and J. J. Knierim. "Hebbian analysis of the transformation of medial entorhinal grid-cell inputs to hippocampal place fields". In: *Journal of neurophysiology* 103.6 (2010), pp. 3167–3183.

[90] D. O. Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[91] A. N. Burkitt. “A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input”. In: *Biological cybernetics* 95.1 (2006), pp. 1–19.

[92] T. Schwalger and A. V. Chizhov. “Mind the last spike—firing rate models for mesoscopic populations of spiking neurons”. In: *Current opinion in neurobiology* 58 (2019), pp. 155–166.

[93] S. Song, K. D. Miller, and L. F. Abbott. “Competitive Hebbian learning through spike-timing-dependent synaptic plasticity”. In: *Nature neuroscience* 3.9 (2000), pp. 919–926.

[94] A. L. Hodgkin and A. F. Huxley. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of physiology* 117.4 (1952), pp. 500–544.

[95] A. L. Hodgkin and A. F. Huxley. “The components of membrane conductance in the giant axon of Loligo”. In: *The Journal of physiology* 116.4 (1952), pp. 473–496.

[96] L. F. Abbott. “Lapicque’s introduction of the integrate-and-fire model neuron (1907)”. In: *Brain research bulletin* 50.5-6 (1999), pp. 303–304.

[97] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[98] B. Si, S. Romani, and M. Tsodyks. “Continuous attractor network model for conjunctive position-by-velocity tuning of grid cells”. In: *PLoS Comput Biol* 10.4 (2014), e1003558.

[99] B. L. McNaughton, F. P. Battaglia, O. Jensen, E. I. Moser, and M.-B. Moser. “Path integration and the neural basis of the’cognitive map’”. In: *Nature Reviews Neuroscience* 7.8 (2006), pp. 663–678.

[100] J. O’Keefe and N. Burgess. “Dual phase and rate coding in hippocampal place cells: theoretical significance and relationship to entorhinal grid cells”. In: *Hippocampus* 15.7 (2005), pp. 853–866.

[101] H. T. Blair, A. C. Welday, and K. Zhang. “Scale-invariant memory representations emerge from moire interference between grid fields that produce theta oscillations: a computational model”. In: *Journal of Neuroscience* 27.12 (2007), pp. 3211–3229.

[102] M. Franzius, R. Vollgraf, and L. Wiskott. “From grids to places”. In: *Journal of computational neuroscience* 22.3 (2007), pp. 297–299.

[103] E. Coumans and Y. Bai. “Pybullet, a python module for physics simulation for games, robotics and machine learning”. In: (2016).

[104] E. Rohmer, S. P. Singh, and M. Freese. “V-REP: A versatile and scalable robot simulation framework”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1321–1326.

[105] V. Braitenberg. *Vehicles: Experiments in synthetic psychology*. MIT press, 1986.

[106] G. S. Fraenkel and D. L. Gunn. “The orientation of animals: kineses, taxes and compass reactions.” In: (1961).

[107] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[108] K. Hardcastle, S. Ganguli, and L. M. Giocomo. "Environmental boundaries as an error correction mechanism for grid cells". In: *Neuron* 86.3 (2015), pp. 827–839.

[109] K. Cheng. "A purely geometric module in the rat's spatial representation". In: *Cognition* 23.2 (1986), pp. 149–178.