Generalized Strategies for Path Integration using Neural Oscillators

by

Xiang Ji

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

Abstract

Path integration is a process by which an animal obtains its location by integrating its velocity over time. Evidence shows that path integration may contribute to certain neural activity patterns in the entorhinal cortex and the hippocampus, which probably serves important function in the animal's navigation system. Such neurons include place cells, firing only when the animal is at a certain region of the environment, and grid cells, which fire when the animal is at locations organized in a hexagonal grid pattern. Various models try to explain the neural mechanism of path integration and how the path integration system serves as the input to grid cells and place cells. Among the best results is the one given by our previous work [28]. The model links Welday et al.'s bank-of-oscillators model [39] with Fourier theory, implements path integration with spiking leaky integrate-and-fire (LIF) neurons, and generates neurons with the desired activity patterns. However, the model depends on a regular placement of its parts. We extend our previous work into a generalized framework by allowing arbitrary placement and introducing a coupling approach that employs an iterative least-squares method. We build a neural network under the new framework with LIF neurons, conduct experiments with various parameter combinations, and evaluate the network's performance with new quantitative measures. Results show clear trends of the performance varying with some of the parameters, which may shed some light on the direction to a more effective path integration system.

Acknowledgments

I would like to express the deepest appreciation to my supervisor, Professor Jeff Orchard. He continually and convincingly conveyed a spirit of adventure in regard to research, and an excitement in regard to teaching. Without his guidance and persistent help this thesis would not have been possible.

I would like to thank the readers of my thesis, Professor Matthijs van der Meer and Professor Gladimir Baranoski, for being willing to spend time and efforts on reviewing my work. Their advice is very important to me.

A thank you to Professor Chris Eliasmith, who laid the foundation of computational neuroscience research in University of Waterloo. His team first proposed the Neural Engineering Framework, on which this thesis is based.

Also, thank my parents, girlfriend and all friends for always supporting me.

Dedication

This is dedicated to my grandfather and maternal grandmother. May they rest in peace.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Spatial Navigation

Spatial navigation, defined as "the process or activity of accurately ascertaining one's position and planning and following a route" [34], is an essential skill of humans and many animals. Examples range from rats exploring mazes [23] to taxi drivers calculating routes to destinations [18].

As the definition implies, spatial navigation is a complex process that involves various sensory input and intricate brain activities. Thus, although the neural mechanisms of spatial navigation have been a hot research topic for years, we still do not thoroughly understand how spatial navigation works. However, experiments and models have shed some light on the neural activities during the spatial navigation process, so that we can speculate on a part of the procedure. The following sections will discuss how space is likely to be represented in the brain, how they velocity input may be incorporated in spatial navigation, and what are the possible approaches to simulate the process from velocity input to spatial representation.

1.2 Cognitive Map

Evidence supports that spatial navigation usually needs some internal representation of the external environment, generally called a "cognitive map" or a "spatial map". The concept was raised as early as the 1930s, when Edward Tolman argued that an animal can form such a map in its brain when it explores the space [36]. However, the argument was more of a guess until we obtained enough knowledge about related neural activities.

Initial evidence was found in the hippocampus. O'Keefe et al. [23] first discovered some special neurons in the rat hippocampus, dubbed *place cells*. A place cell fires only when the animal is located in a specific small area called the *firing field*. They then argued that the hippocampus is where the internal cognitive map is used for navigation [25].

Findings of other cell types reveal possible origins of such cognitive maps. Hafting et al. [12] reported cells in the rat medial entorhinal cortex (MEC) that fire when a rat is in areas forming a hexagonal grid pattern. These cells are usually called *grid cells*. Sargolini et al. [31] found cells in the MEC that fire when the animal's head is oriented toward a certain direction relative to the environment (*head direction cells*). In the same brain region, Solstad et al. [32] discovered neurons that react when the animal is near the border of the environment (*border cells*). Considering the strong association between these cell activities and the animal's position and direction, as well as the MEC being a main cortical input to the hippocampus [14], it is believed that these cells may play important roles in the animal's internal spatial representation.

Details of place cells and grid cells continue to be discovered. Muller et al. [21] first reported that place cell firing patterns change with the environment. It was observed that different subsets of place cells are involved in different surroundings. O'Keefe et al. [26] discovered that the spike timing of place cells is related to the phase of the theta wave (a strong oscillation that can be observed in the hippocampal region), forming a phenomenon called *phase precession*. The firing usually begins at a particular phase when the rat enters a place field, moving forward on each theta cycle during the traversal. It was argued that phase precession is used to achieve more accurate place encoding [26], and may reflect interference among neural oscillators [10, 17]. It is also found that place cell firing fields persist when the rat is in darkness, indicating that the input to place cells should contain more than just visual information [30]. Grid cells show a similar ability to continue their firing patterns in dark surroundings [12]. Unlike place cells, however, the same set of grid cells keep firing in different environments, with grid patterns rescaled when surrounded walls are relocated [33]. Moreover, Stensola et al. [33] showed that grid cells are clustered, forming anatomically overlapping modules with distinct grid patterns. The firing patterns of cells within each module are similar. It was also reported that grid size increases in general from dorsal to ventral MEC [12].

1.3 Path Integration

The spatial navigation system utilizes various information as input, among which the animal's velocity serves as an important part. Even without sensory input, an animal can estimate how far it has moved by integrating its velocity, a process called path integration [38]. The fact that many animals are able to locate the goal sites only by relying on self-motion cues makes it convincing that path integration serves as an import factor in the navigation system [8]. An famous experiment that demonstrates path integration is the *Morris water maze* [20], in which a rat is placed in a circular pool and is supposed to find an invisible platform that allows it to escape the water (Figure 1.1 (a)). Evidence shows that the rat is able determine the position of the platform using the information of its velocity, which shows the existence of path navigation in the navigation process (Figure 1.1 (b)). While the space-related firing patterns may also be generated from other mechanisms like cue associations, triangulation or landmark vector navigation [22], evidences support that the path integration system serves as an important input to place cells and grid cells, since these neurons are able to keep their firing patterns even in dark surroundings [12, 19].

1.4 Models

Quite a few models have been raised, trying to explain how path integration happens and how those position-related cell activities can be generated from path integration. These models shed some light on how to achieve the path integration function while preserving the biological plausibility, so that they provide persuasive hypotheses about how path integration is actually performed in the animal's brain. Successful models basically fall into two categories. One category is called *attractor network models*, in which neural activity patterns exhibit stable states (attractors), corresponding to local minima of an energy function. Noise helps the system settle into a global minimum energy state. With appropriate design, the global minimum energy state can be a function of the animal's position, while superimposing different global minima gives different position-related firing patterns [9]. The other category is called *oscillatory interference models*. These models

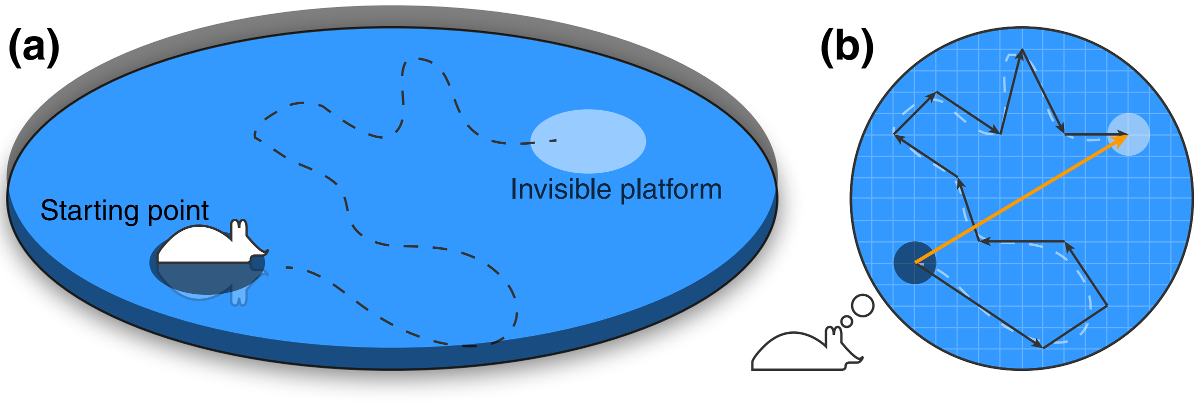

Figure 1.1: A Morris water maze experiment. **(a)** A rat is put in a circular water pool (black ellipse), so that it needs to keep swimming until it reaches a platform that is hidden under water (white ellipse). An example trajectory is shown as the dashed line. **(b)** The rat may sum up velocity vectors (black) in order to keep track of its current location (orange), exhibiting a path integration process.

employ *velocity-controlled neural oscillators* (VCOs), composed of either single neuron membrane oscillations [24] or neuron ensemble activities [41]. The oscillating frequency of a VCO is related to the animal's velocity, thus the animal's position can be encoded in the VCO phase. Position-related cell activities emerge from the interference of certain VCO oscillations.

Both classes of models have explanations for some experimental findings, and keep evolving as more facts are reported. My collaborators and I recently published a framework [28], which links Fourier theory to oscillatory interference models. Based on Welday et al.'s work [39], we suggested that by arranging VCOs on a 2-D plane according to their velocity gains, the VCO phases should constitute a linear relationship, of which the slope indicates the animal's position. We found that it has a similar form to the Fourier Shift Theorem, and assigning weights to different VCOs is like assigning Fourier coefficients to different Fourier basis functions. Thus we argued that by using weighted VCO phases as input, we can build neurons with any shape of firing fields. We then raised a model using three arrays of VCOs with coupling mechanisms that reduce the influence of noise. The model successfully served as the input to various place cells, grid cells and border cells, as well as exhibited theta phase precession.

1.5 Thesis Objectives

Working well in experiments, though, our previous model still leaves space for improvement. In our previous model, both VCOs and coupling neurons need to be set up in a rather contrived manner. While such structures may actually exist in brains, it is interesting to study models with a more generalized VCO arrangement and coupling scheme. By easing some of the restrictions, our work expands the choice of parameters in oscillatory interference models, which allows further studies on how parameter change could possibly affect the models' behavior. Those studies aiming at understanding how the entorhinal cortex and the hippocampus encode and compute with space would add to our emerging understanding of how the brain works.

In the following chapters, we will explain our previous model in detail, reform the theory to allow arbitrary VCO placement, and raise a network with random VCO arrangement and corresponding coupling mechanisms, followed by experimental results and further discussions.

Chapter 2

Background

2.1 Velocity-Controlled Oscillator

An oscillator is a system in which a *phase vector*

$$\boldsymbol{s}(t) = \begin{bmatrix} s_x(t) \\ s_y(t) \end{bmatrix} , \tag{2.1}$$

usually with the magnitude of 1, keeps rotating over time (Figure 2.1). We refer to the angle between the phase vector and the x-axis as the *phase* of an oscillator. The relationship between the phase $\phi(t)$ and the phase vector (written as a complex number) is,

$$e^{i\phi(t)} = \cos\phi(t) + i\sin\phi(t) = s_x(t) + is_y(t) \ . \tag{2.2}$$

We call an oscillator a "velocity-controlled oscillator" (VCO) when its frequency is modulated by the animal's velocity. Here we use linear VCOs, whose frequency is a linear function of the animal's velocity.

In 2-D environments, the frequency of a linear VCO can be modeled as

$$\omega_i(t) = c_{i,x}v_x(t) + c_{i,y}v_y(t) + \omega_b \ . \tag{2.3}$$

Here $i \in \{1, 2, \ldots, n\}$ indicates the index of the VCO (suppose we have n VCOs). The oscillating frequency of VCO i at time t is represented by $\omega_i(t)$. The animal's velocity

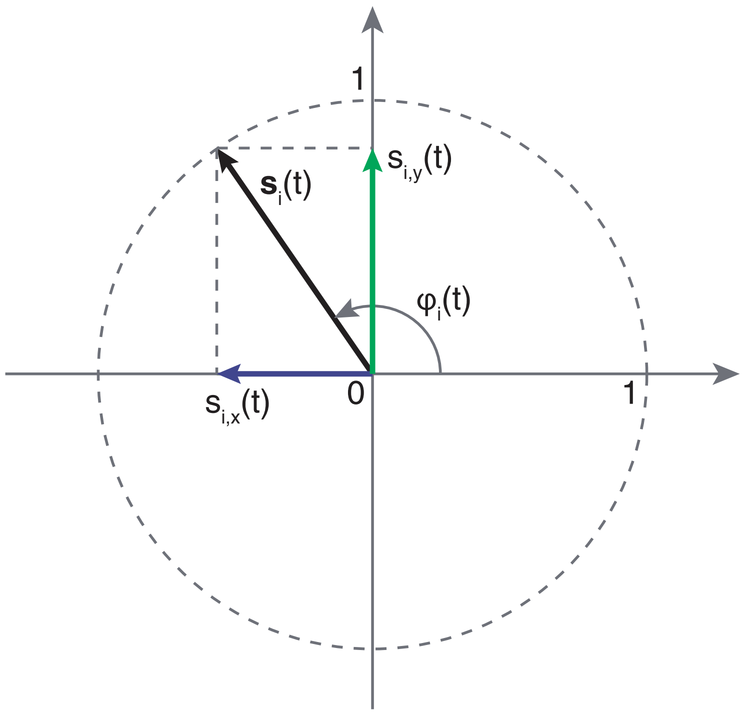

Figure 2.1: Demonstration of a phase vector. The encoded values are $s_{i,x}(t)$ and $s_{i,y}(t)$, while the angle between the phase vector and the x-axis represents the VCO phase.

components along the x- and y-axis at time t are $v_x(t)$ and $v_y(t)$. Parameters $c_{i,x}$ and $c_{i,y}$ determine how the velocity affects $\omega_i(t)$. We name the 2-D vector

$$\begin{bmatrix} c_{i,x} & c_{i,y} \end{bmatrix} \tag{2.4}$$

the VCO's *address*, so that each VCO can be represented as a point in an 2-D *address space*. The base oscillating frequency of VCOs is ω_b, remaining invariant among different VCOs.

From Equation 2.3 we can see that the frequency of a VCO is a linear combination of the velocity's x-component and y-component. Because of this, we can prove that VCO phases are also linear with the animal's position. Since phase is the integral of frequency over time, we have

$$\phi_i(t) = \int_0^t \omega_i(\tau)\,\mathrm{d}\tau \ , \tag{2.5}$$

in which $\phi_i(t)$ is the phase of VCO i at time t. Considering the expansion of $\omega_i(\tau)$ and integrating the terms separately, we have

$$\phi_i(t) = c_{i,x}x(t) + c_{i,y}y(t) + \phi_b(t) \ , \tag{2.6}$$

in which

$$\begin{cases} x(t) = \displaystyle\int_0^t v_x(\tau)\,\mathrm{d}\tau \\ y(t) = \displaystyle\int_0^t v_y(\tau)\,\mathrm{d}\tau \\ \phi_b(t) = \displaystyle\int_0^t \omega_b\,\mathrm{d}\tau \end{cases} \quad . \tag{2.7}$$

Here we notice that $x(t)$, the integral of $v_x(\tau)$ over time, is the current x-position. In the same way, the current y-position is indicated by $y(t)$, which indicates that VCO phases relate linearly to positions.

That is, the phase of each VCO is dictated by Equation 2.6, according to its address in the space, and the animal's location. Consider the address space, the 2-D domain of $[c_x\ c_y]$ coordinates. From Equation 2.6, we can see that VCO phase is a linear function over that domain, and forms a plane. We call this the *phase ramp*. Notice that the slope (gradient) of the phase ramp is $[x(t)\ y(t)]$. In this way, the slope of the phase ramp represents the animal's position in space.

2.2 Previous Model

If we have perfect VCOs, we can simply use them for path integration without extra effort. However, this is not the case. When modeling VCOs with neurons, the intrinsic neural noise tends to cause VCO phases to drift from their ideal values, which often confounds the path integration process. In our previous paper [28] (as well as in [13, 29]), we raise a model that can effectively perform path integration tasks despite this noise.

The model contains four levels (Figure 2.2 (a)). The bottom level is a velocity node, providing input to all VCOs. The second level incorporates three VCO propellers, each containing 17 VCOs. Their arrangement in the address space is shown in Figure 2.2 (b). The VCO phases in each propeller should form a linear ramp. The propellers are arranged at angles 0°, 120° and 240° in the address space. The 9th VCO in each propeller is placed at the origin. The distance between adjacent VCOs in each propeller is equal. VCOs receive input from the velocity node, so that their phases encode the animal's position as described in the previous section.

The third level includes three propellers, each with 16 phase-step nodes. As previously discussed, neural activities tend to be noisy, so when we build VCOs using neurons, the VCO phases will deviate from their ideal values. However, the deviation can be inhibited with multiple VCOs. If the VCO phase drifts are uncorrelated, we should still be able to get an accurate estimate of the slope of the phase ramp, which should deviate less from the ideal than the phase of a single VCO. Using that slope, we then suppress the VCO phase noise by forcing the phases to approach the estimated phase ramp. In this model, we achieve this coupling by using the three propellers of phase-step nodes in the third level.

Phase-step nodes can keep VCO phases near a linear ramp in each propeller. Each phase-step propeller corresponds to a VCO propeller, so that each phase-step node couples two adjacent VCOs (Figure 2.2 (c)). Since VCOs are evenly distributed in each propeller, the phase difference of any two adjacent VCOs should be equal. To ensure the equality, each phase-step node computes the phase difference for the corresponding VCO pair and broadcasts it to all the other phase-step nodes for that propeller, so that they can reach a consensus. Finally each phase-step node compares the consensus with the local version of the phase difference, and sends the correction back to its VCO pair (Figure 2.3).

Even if the phases of VCOs in each propeller form a nice linear ramp, the three propellers might not be coplanar. We need another level of coupling to keep VCOs in phase

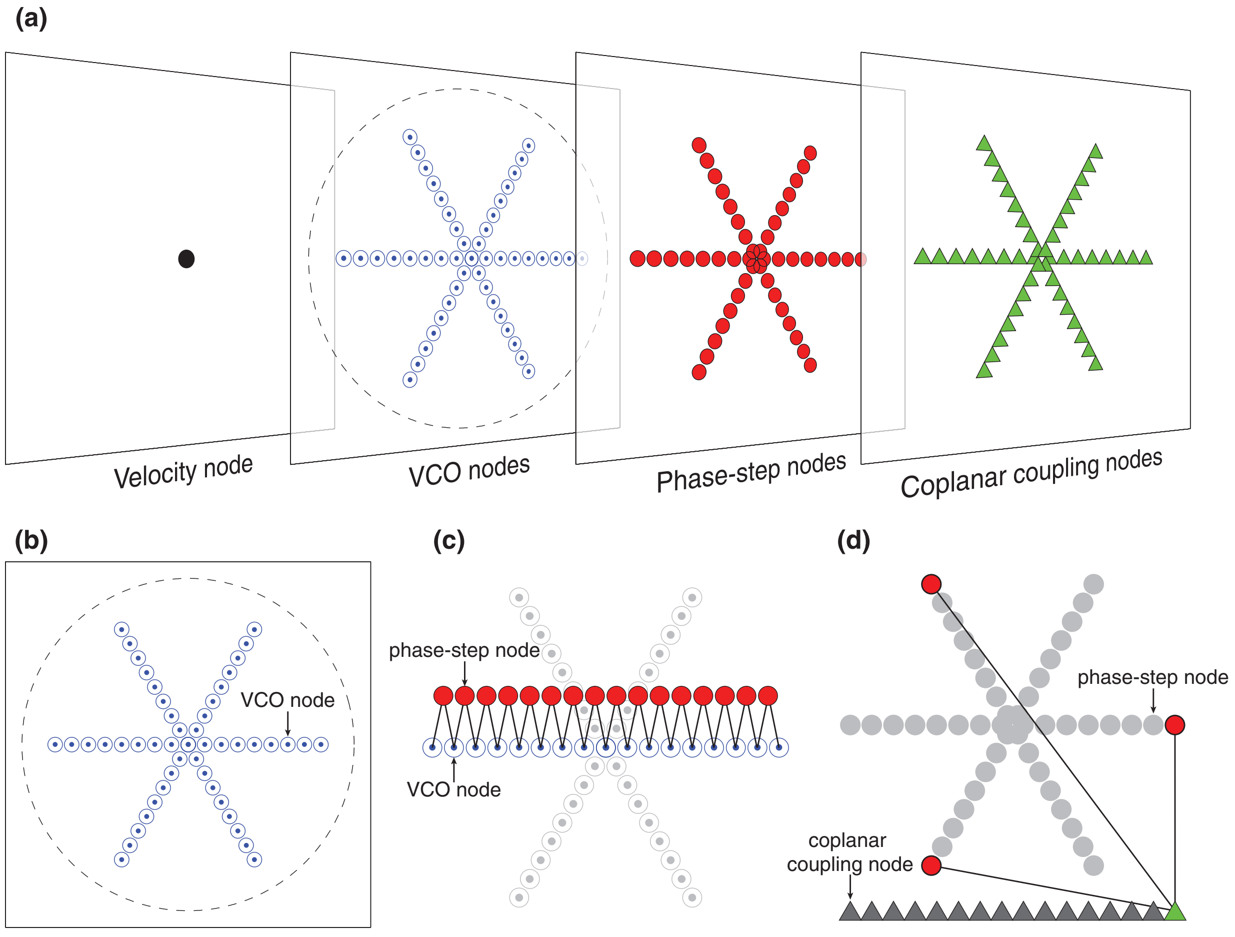

Figure 2.2: Our previous model. **(a)** Network layers. **(b)** Arrangement of VCOs (circles with centered dots) in the address space. Three propellers are shown, each with 17 VCOs. **(c)** Phase-step node coupling. The phase-step propeller at angle 0° (filled circles) is shown as an example. Each phase-step node connects to two adjacent VCOs. Phase step nodes in the same propeller are randomly connected with each other, which is not displayed in the figure. **(d)** Coplanar coupling. The coplanar coupling propeller at angle 0° (triangles) is shown as an example. Each coplanar node connects with three phase-step nodes, equidistant from the origin.

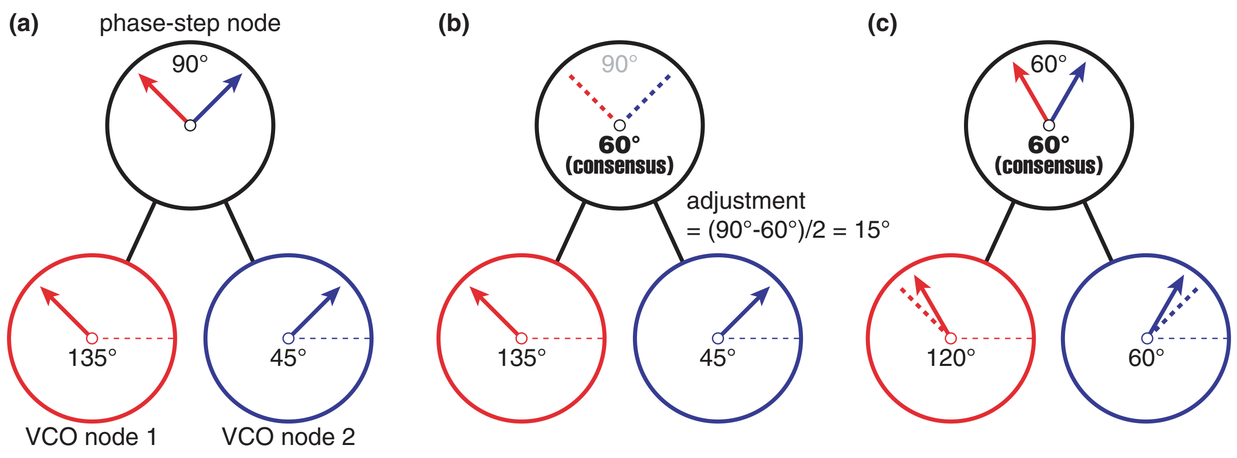

Figure 2.3: VCO coupling with phase-step nodes. **(a)** Two VCOs are at phase 135° and 45°. The phase-step node computes their phase difference. **(b)** All phase step nodes reach a consensus that the average phase difference should be 60°. It infers that the adjustment to each VCO should be 15°. **(c)** The adjustment is sent back to VCOs.

across propellers. A coplanar coupling node in this level connects with three phase-step nodes, as shown in Figure 2.2 (d). The three phase-step nodes have an interesting property; since they are equidistant from the origin at angles 0°, 120° and 240°, the sum of the three phase differences should be 0. Because of this, a coplanar coupling node can keep the three phase-step nodes in a plane by adding up the three phase differences, comparing the value with 0, and sending the correction back to the phase-step node it corresponds to. The fourth level comprises three propellers, each with 16 such coplanar coupling nodes. Each coplanar coupling propeller corresponds to a phase-step propeller. Inside the propeller, each coplanar coupling node maps to a different phase-step node.

Experiments show that this model is able to perform effective path integration for several seconds when implemented with spiking leaky integrate-and-fire (LIF) neurons [5, 9, 15].

To achieve long-lasting path integration, real organisms seem to use sensory input to combat phase drift in the path integration system [3, 40]. Inspired by other models incorporating sensory input [3, 5], we also previously proposed a mechanism for sensory input to correct the oscillator phases, which exhibits effective inhibition of VCO phase drift [13].

2.3 Oscillatory Interference and Fourier Theory

The VCOs oscillate, so various combinations of them can result in complex interference patterns. Since VCO phases encode the animal's position, an interference pattern can be seen as a function of the animal's position laid out over the animal's environment. When using a VCO combination as the input to a certain neuron, that neuron's activity should also be a function of the animal's position, so that it is possible to generate cells with different firing fields by connecting it to different VCOs with different connection weights. But how can we determine the connection scheme to obtain a neuron with a desired firing field? Our previous work [28] gives a possible answer by linking Fourier theory to the oscillatory interference model.

Fourier theory tells us that, under appropriate conditions, an arbitrary function $f(x)$ can be mapped to a function $\hat{f}(\hat{x})$ in a different parameter space with the help of a group of complex Fourier basis functions. The mapping takes the form

$$\hat{f}(\hat{x}) = \int_{-\infty}^{\infty} f(x) e^{2\pi i \hat{x} x} \, \mathrm{d}x \ , \tag{2.8}$$

in which $e^{2\pi i \hat{x} x} = \cos(2\pi \hat{x} x) + i \sin(2\pi \hat{x} x)$ is a Fourier basis function. The mapping is called the Fourier transform (FT). We can also map $\hat{f}(\hat{x})$ back to $f(x)$ through the inverse Fourier transform (IFT),

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\hat{x}) e^{-2\pi i \hat{x} x} \, \mathrm{d}\hat{x} \ . \tag{2.9}$$

In 2-D, the FT and IFT can be written as

$$\hat{f}(\hat{x}, \hat{y}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{2\pi i (\hat{x} x + \hat{y} y)} \, \mathrm{d}x \mathrm{d}y \ , \tag{2.10}$$

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{f}(\hat{x}, \hat{y}) e^{-2\pi i (\hat{x} x + \hat{y} y)} \, \mathrm{d}\hat{x} \mathrm{d}\hat{y} \ . \tag{2.11}$$

How can we link firing fields to VCO phases using the Fourier transform? First we argue that VCO phases can be transformed into a group of Fourier basis functions. Setting the phase of the VCO at the origin as the frame of reference, denoted ϕ_b, Equation 2.6 tells us that the relative VCO phase $\phi - \phi_b$ is a function of both the animal's position and the VCO address, namely

$$\phi - \phi_b = c_x x + c_y y \ . \tag{2.12}$$

Given that relative phase, the corresponding phase vector can be written as a Fourier basis function (complex exponential),

$$e^{2\pi i(\phi-\phi_b)} = e^{2\pi i(c_x x + c_y y)} \ . \tag{2.13}$$

Figure 2.4 shows one component of the phase vector as a function of (x, y) location.

Then we note that a firing field can be treated as a scalar function in the animal's spatial environment by representing neural activity levels with numeric values. On the other hand, with a read-out node receiving the output from each VCO, the connection weights from VCOs to the read-out node can be considered as a function in the address space. Denoting the firing field function as $a(x, y)$, we can find the corresponding connection weight function $w(c_x, c_y)$ in the address space following Equation 2.11, namely

$$a(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(c_x, c_y)e^{2\pi i(\phi-\phi_b)}\,\mathrm{d}c_x\mathrm{d}c_y \ , \tag{2.14}$$

and

$$w(c_x, c_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a(x, y)e^{-2\pi i(\phi-\phi_b)}\,\mathrm{d}x\mathrm{d}y \ . \tag{2.15}$$

Notice that in Equation 2.14, the firing field function is represented exactly by adding up a weighted sum of all the VCO phase vectors. Now we are able to theoretically construct an arbitrary firing field using VCO phases and the corresponding connection weight function. It is just the IFT of the firing field function.

In practice, instead of integrating over the whole address space, we can approximate the firing field by sampling the address space with n VCOs, so that Equation 2.14 is replaced by

$$a(x, y) \approx \sum_{j=1}^{n} w(c_{j,x}, c_{j,y})e^{2\pi i(\phi_j-\phi_b)} \ . \tag{2.16}$$

By connecting a readout node to the set of all VCOs using weights $w(c_{j,x}, c_{j,y}),\ j \in \{1, \cdots, n\}$, the node should exhibit an activity pattern similar to $a(x, y)$ (Figure 2.5). In this approach, experiments successfully generated the activation patterns of grid cells using spking LIF neurons, as shown in Figure 2.6.

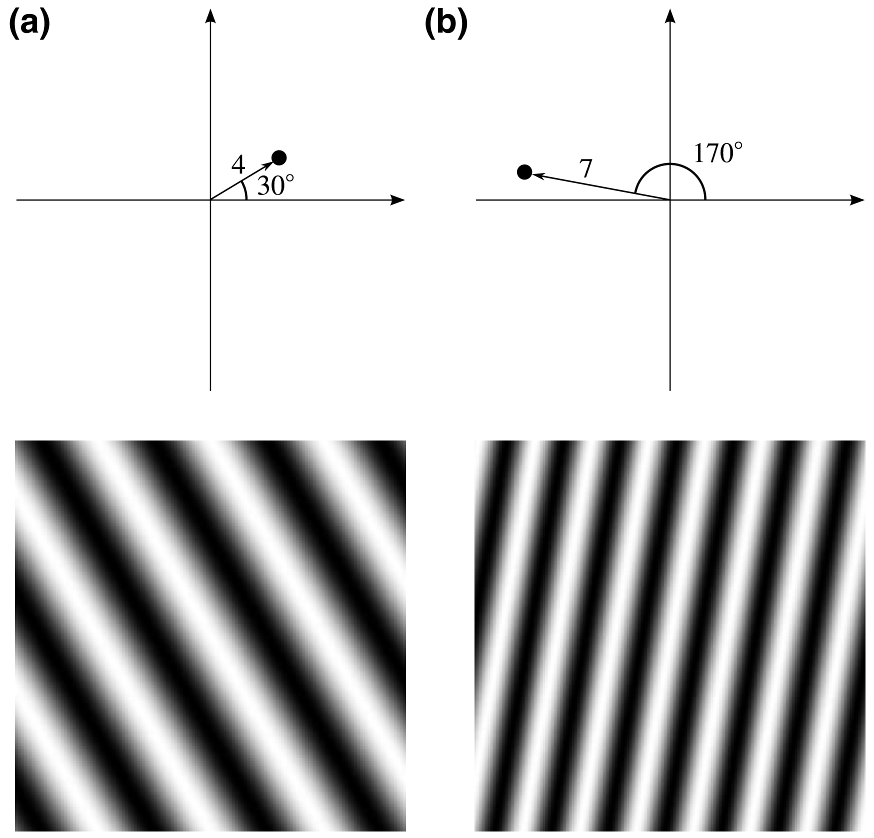

Figure 2.4: Demonstration of Fourier basis functions. The top row shows two examples in the address space. **(a)** The address is at 30°, 4 units from the origin. **(b)** The address is at 170°, 7 units from the origin. The bottom row shows the real part of the corresponding Fourier basis function with c_x and c_y set as the address above. Taken with permission from [28].

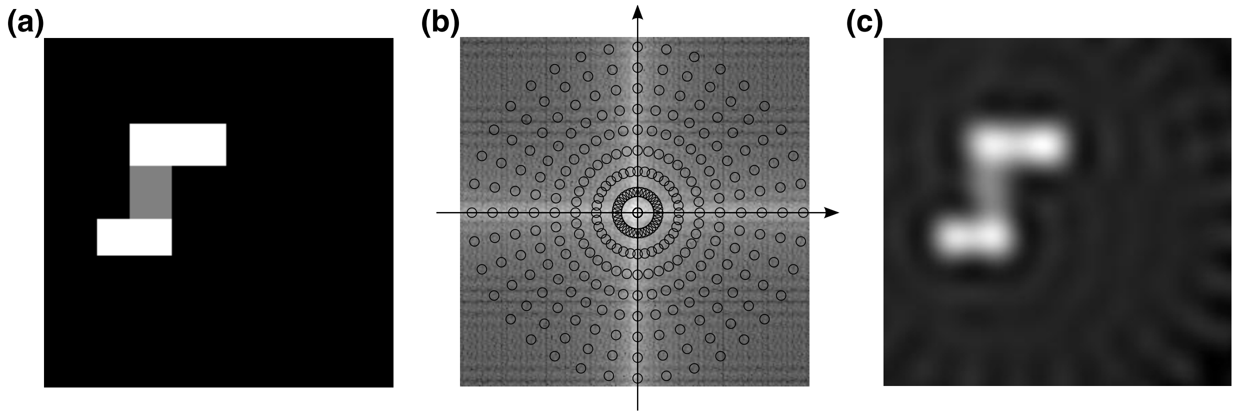

Figure 2.5: Example of firing field reconstruction. **(a)** The ideal firing field $a(x, y)$. **(b)** The modulus of $w(c_x, c_y)$, overlaid with VCO locations (18 propellors, 9 rings). **(c)** The firing field resulting from combining all the weighted Fourier basis functions corresponding to the VCOs in **(b)**. Taken with permission from [28].

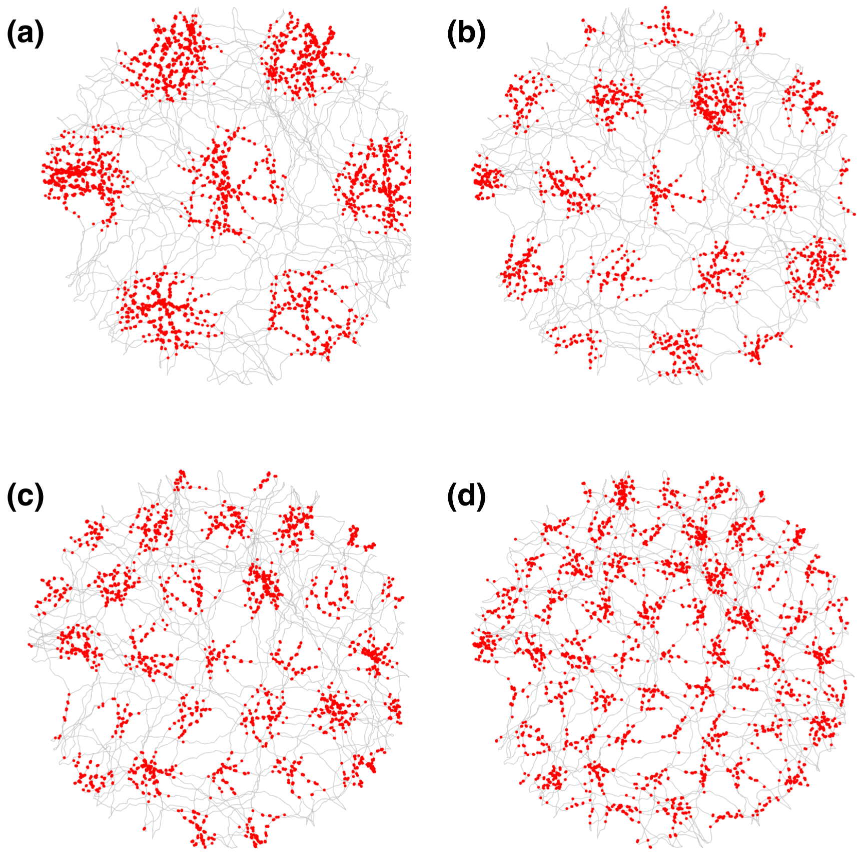

Figure 2.6: Example grid cell firing patterns. Different connection weights result in different grid sizes. Taken with permission from [28].

Chapter 3

Generalized Models

3.1 Path Integration with Arbitrarily Placed VCOs

Based on our previous work, we realize that the 3-propeller model is a special case to achieve oscillatory interference path integration. The placement of the VCOs in the address space is highly regular, and the coupling scheme is rather contrived. We wish to establish a more generalized framework that eases some of the restrictions, and to observe how this liberalization will affect the system's behavior.

Different from what is stated in our previous work, we do not specify how VCOs are arranged in the address space. Instead, we choose $c_{i,x}$ and $c_{i,y}$ arbitrarily.

As in the 3-propeller model, we start from the relationship between VCO phases and the animal's position (Equation 2.6), namely

$$\phi_i(t) = c_{i,x}x(t) + c_{i,y}y(t) + \phi_b(t) \ .$$

We can express this phase-position relation for multiple VCOs using matrix notation,

$$\begin{bmatrix} \phi_1(t) \\ \phi_2(t) \\ \vdots \\ \phi_n(t) \end{bmatrix} = \begin{bmatrix} c_{1,x} & c_{1,y} & 1 \\ c_{2,x} & c_{2,y} & 1 \\ \vdots & \vdots & \vdots \\ c_{n,x} & c_{n,y} & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ \phi_b(t) \end{bmatrix} \ . \tag{3.1}$$

In short, we have

$$\boldsymbol{\phi}(t) = \boldsymbol{C}\boldsymbol{p}(t) \ , \tag{3.2}$$

in which

$$\boldsymbol{\phi}(t) = \begin{bmatrix} \phi_1(t) \\ \phi_2(t) \\ \vdots \\ \phi_n(t) \end{bmatrix} \ , \tag{3.3}$$

$$\boldsymbol{C} = \begin{bmatrix} c_{1,x} & c_{1,y} & 1 \\ c_{2,x} & c_{2,y} & 1 \\ \vdots & \vdots & \vdots \\ c_{n,x} & c_{n,y} & 1 \end{bmatrix} \ , \tag{3.4}$$

and

$$\boldsymbol{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \phi_b(t) \end{bmatrix} \ . \tag{3.5}$$

Noticing that the animal's position, $x(t)$ and $y(t)$, are two components contained in vector $\boldsymbol{p}(t)$, we find that Equation 3.2 relates the animal's position to VCO phases over time. The equation illustrates that, as the animal moves around, the phases reflect that motion through a linear relationship. In fact, when drawn in the 2-D address space, where each VCO is drawn at its address (c_x, c_y), the phases form a plane, and the slope (gradient) of that plane is numerically equal to the animal's location, $(x(t), y(t))$.

Here, we derive a universal approach that estimates the slope of the phase ramp for arbitrarily arranged VCOs. Suppose that we have m VCO pairs selected from n VCOs, in which pair k contains VCOs with indices i_k and j_k ($k \in \{1, 2, \ldots, m\}$, $i_k, j_k \in \{1, 2, \ldots, n\}$, $i_k \neq j_k$). Then the phase difference in pair k can be denoted as

$$\begin{aligned} \Delta\phi_k(t) &= \phi_{i_k}(t) - \phi_{j_k}(t) \\ &= \Big(c_{i_k,x}x(t) + c_{i_k,y}y(t) + \phi_b(t)\Big) - \Big(c_{j_k,x}x(t) + c_{j_k,y}y(t) + \phi_b(t)\Big) \\ &= (c_{i_k,x} - c_{j_k,x})x(t) + (c_{i_k,y} - c_{j_k,y})y(t) \\ &= \Delta c_{k,x}x(t) + \Delta c_{k,y}y(t) \ , \end{aligned} \tag{3.6}$$

in which

$$\begin{aligned} \Delta c_{k,x} &= c_{i_k,x} - c_{j_k,x} \ , & (3.7) \\ \Delta c_{k,y} &= c_{i_k,y} - c_{j_k,y} \ . & (3.8) \end{aligned}$$

Here we notice that we can use the phase **differences** and the VCO addresses to estimate the slope. Similar to previous matrix representation, we have

$$\Delta \boldsymbol{\phi}(t) = \Delta \boldsymbol{C} \boldsymbol{p}'(t) \ , \tag{3.9}$$

in which

$$\Delta \boldsymbol{\phi}(t) = \begin{bmatrix} \Delta\phi_1(t) \\ \Delta\phi_2(t) \\ \vdots \\ \Delta\phi_m(t) \end{bmatrix} \ , \tag{3.10}$$

$$\Delta \boldsymbol{C} = \begin{bmatrix} \Delta c_{1,x} & \Delta c_{1,y} \\ \Delta c_{2,x} & \Delta c_{2,y} \\ \vdots & \vdots \\ \Delta c_{m,x} & \Delta c_{m,y} \end{bmatrix} \ , \tag{3.11}$$

and

$$\boldsymbol{p}'(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \ . \tag{3.12}$$

Reconstructing the slope, or position, from phase differences is equivalent to computing $\boldsymbol{p}'(t)$ when $\Delta\boldsymbol{\phi}(t)$ and $\Delta\boldsymbol{C}$ are known. However, the linear system in Equation 3.9 is over-determined (more equations than unknowns). We can find the least-squares solution of this system by multiplying the pseudo-inverse of $\Delta\boldsymbol{C}$ on both sides of Equation 3.9, which gives

$$\boldsymbol{p}'(t) = (\Delta\boldsymbol{C}^{\mathrm{T}}\Delta\boldsymbol{C})^{-1}\Delta\boldsymbol{C}^{\mathrm{T}}\Delta\boldsymbol{\phi}(t) \ . \tag{3.13}$$

Using this method, we are able to decode the animal's position from the phase differences among a collection of VCO pairs and their relative displacements in the address space. We will discuss how this solution is implemented into our neural network later in the thesis.

3.2 Phase Coupling on Arbitrarily Placed VCOs

With arbitrarily arranged VCOs, we still need to solve the problem that imperfect oscillators tend to drift out of phase. A coupling scheme is necessary, but it should be different from that in our previous work, since we do not expect equal phase differences from arbitrarily placed VCOs.

Looking back at Equation 3.9, we notice that drift in a single VCO will cause $\Delta\boldsymbol{\phi}(t)$ to deviate from $\Delta\boldsymbol{C}\boldsymbol{p}'(t)$. In other words, a coupling scheme that suppresses drift needs to correct $\Delta\boldsymbol{\phi}(t)$ and $\boldsymbol{p}'(t)$ so that the difference between $\Delta\boldsymbol{\phi}(t)$ and $\Delta\boldsymbol{C}\boldsymbol{p}'(t)$ is reduced.

Naturally, we evaluate their difference using the 2-norm of the residual, namely

$$\begin{aligned} E\Big(\Delta\boldsymbol{\phi}(t),\, \boldsymbol{p}'(t)\Big) &= \Big|\Big|\Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t)\Big|\Big|^2 \\ &= \Big(\Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t)\Big)^{\mathrm{T}}\Big(\Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t)\Big)\,. \end{aligned} \tag{3.14}$$

To minimize $E\left(\Delta\boldsymbol{\phi}(t),\, \boldsymbol{p}'(t)\right)$, we consider employing the gradient descent method. First we compute gradient vectors of $E\left(\Delta\boldsymbol{\phi}(t),\, \boldsymbol{p}'(t)\right)$ with respect to both $\Delta\boldsymbol{\phi}(t)$ and $\boldsymbol{p}'(t)$. The gradient vector with respect to $\Delta\boldsymbol{\phi}(t)$ is

$$\begin{aligned} \frac{\partial E\Big(\Delta\boldsymbol{\phi}(t),\, \boldsymbol{p}'(t)\Big)}{\partial \Delta\boldsymbol{\phi}(t)} &= 2\Delta\boldsymbol{\phi}(t) - 2\Delta\boldsymbol{C}\boldsymbol{p}'(t) \\ &= 2\Big(\Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t)\Big)\,, \end{aligned} \tag{3.15}$$

while the gradient vector with respect to $\boldsymbol{p}'(t)$ is

$$\begin{aligned} \frac{\partial E\Big(\Delta\boldsymbol{\phi}(t),\, \boldsymbol{p}'(t)\Big)}{\partial \boldsymbol{p}'(t)} &= -2\Delta\boldsymbol{C}^{\mathrm{T}}\Delta\boldsymbol{\phi}(t) + 2\Delta\boldsymbol{C}^{\mathrm{T}}\Delta\boldsymbol{C}\boldsymbol{p}'(t) \\ &= -2\Delta\boldsymbol{C}^{\mathrm{T}}\Big(\Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t)\Big)\,. \end{aligned} \tag{3.16}$$

We reduce $E\left(\Delta\boldsymbol{\phi}(t),\, \boldsymbol{p}'(t)\right)$ by shifting $\Delta\boldsymbol{\phi}(t)$ and $\boldsymbol{p}'(t)$ in the direction opposite to the gradient vector. We update $\Delta\boldsymbol{\phi}(t)$ using

$$\begin{aligned} \Delta\boldsymbol{\phi}(t) &\leftarrow \Delta\boldsymbol{\phi}(t) - \frac{\gamma}{2}\frac{\partial E\Big(\Delta\boldsymbol{\phi}(t),\, \boldsymbol{p}'(t)\Big)}{\partial \Delta\boldsymbol{\phi}(t)} \\ &= \Delta\boldsymbol{\phi}(t) - \gamma\Big(\Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t)\Big)\,. \end{aligned} \tag{3.17}$$

We define the error vector $\boldsymbol{e}(t)$ as

$$\boldsymbol{e}(t) = \Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t) \; . \tag{3.18}$$

Then we can write the update rule as

$$\Delta\boldsymbol{\phi}(t) \leftarrow \Delta\boldsymbol{\phi}(t) - \gamma\boldsymbol{e}(t) \; . \tag{3.19}$$

Similarly, we update $\boldsymbol{p}'(t)$ as

$$\begin{aligned} \boldsymbol{p}'(t) &\leftarrow \boldsymbol{p}'(t) - \frac{\gamma}{2}\frac{\partial E\Big(\Delta\boldsymbol{\phi}(t),\, \boldsymbol{p}'(t)\Big)}{\partial \boldsymbol{p}'(t)} \\ &= \boldsymbol{p}'(t) + \gamma\Delta\boldsymbol{C}^{\mathrm{T}}\Big(\Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t)\Big) \\ &= \boldsymbol{p}'(t) + \gamma\Delta\boldsymbol{C}^{\mathrm{T}}\boldsymbol{e}(t) \; . \end{aligned} \tag{3.20}$$

Here γ is the parameter controlling the step size.

Equations 3.19 and 3.20 give the basic idea of how to update VCO phases and the phase ramp slope to reduce drifts with arbitrarily arranged VCOs. Three vectors need to be stored by neurons, namely $\Delta\boldsymbol{\phi}(t)$, $\boldsymbol{p}'(t)$ and $\boldsymbol{e}(t)$. In the following sections we will discuss how to practically encode these values using spiking neurons.

3.3 Network Architecture

In this section we present a design of a path integration system following the theory above that exhibits good performance when simulated with spiking LIF neurons. Implementing neural networks using spiking LIF neurons needs extra consideration to overcome the realities, like spiking noise and the limited range of neural firing rates. Appendix A describes the Neural Engineering Framework (NEF) that our network is based on, and how spiking neurons can be used to encode and transform data, and model dynamic processes. In this section, we discuss the details of implementing our path-integration system using the NEF.

3.3.1 Network Overview

The whole network is composed of four types of nodes: a *velocity node*, *VCO nodes*, *coupler nodes* and a *slope node*. Typcically such a network contains one velocity node and

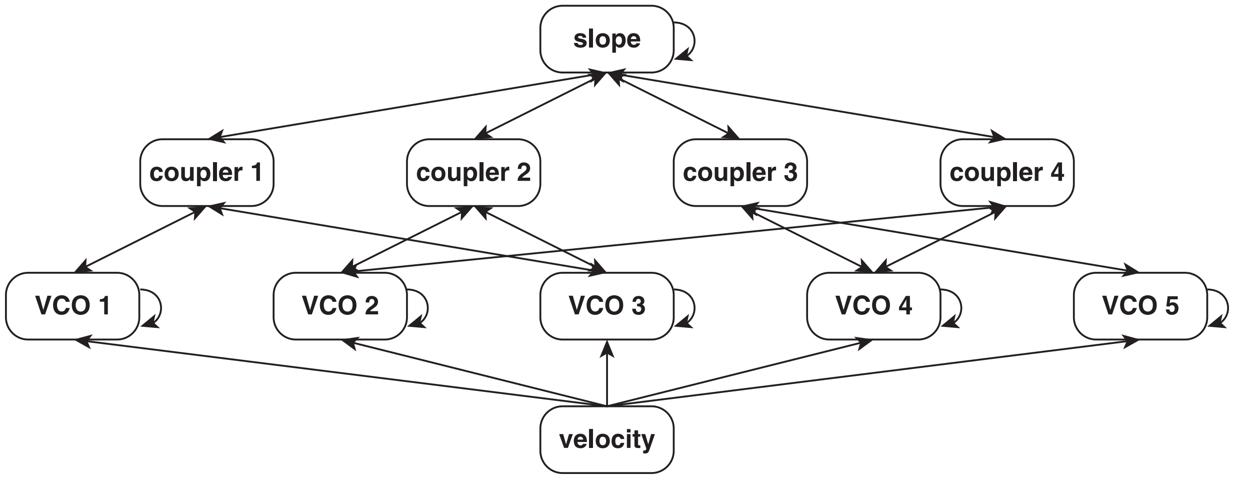

Figure 3.1: An example network architecture. The network contains a velocity node, five VCO nodes, four coupler nodes and a slope node. The velocity node connects to all VCO nodes. Each coupler node connects with two VCO nodes, and a VCO node can be linked to any number of coupler nodes. All coupler nodes are connected with the slope node. All VCO nodes and the slope node have recurrent connections.

one slope node, but multiple VCO nodes and coupler nodes. Figure 3.1 shows a network with a velocity node, five VCO nodes, four coupler nodes and a slope node.

3.3.2 Velocity Node

The velocity node serves as the input of the system. It has two dimensions, corresponding to $v_x(t)$ and $v_y(t)$, as shown in Figure 3.2 (a). It connects to all VCO nodes, each with different weights. For example, VCO i will receive the input

$$v_i(t) = c_{i,x}v_x(t) + c_{i,y}v_y(t) \ , \tag{3.21}$$

which is the velocity controlled part of the VCO frequency $\omega_i(t)$, as shown in Equation 2.3

$$\omega_i(t) = c_{i,x}v_x(t) + c_{i,y}v_y(t) + \omega_b \ .$$

Recall from Equation 2.6 (repeated below as Equation 3.22) that these frequencies result in the linear relationship between VCO phases and the animal's position,

$$\phi_i(t) = c_{i,x}x(t) + c_{i,y}y(t) + \phi_b(t) \ . \tag{3.22}$$

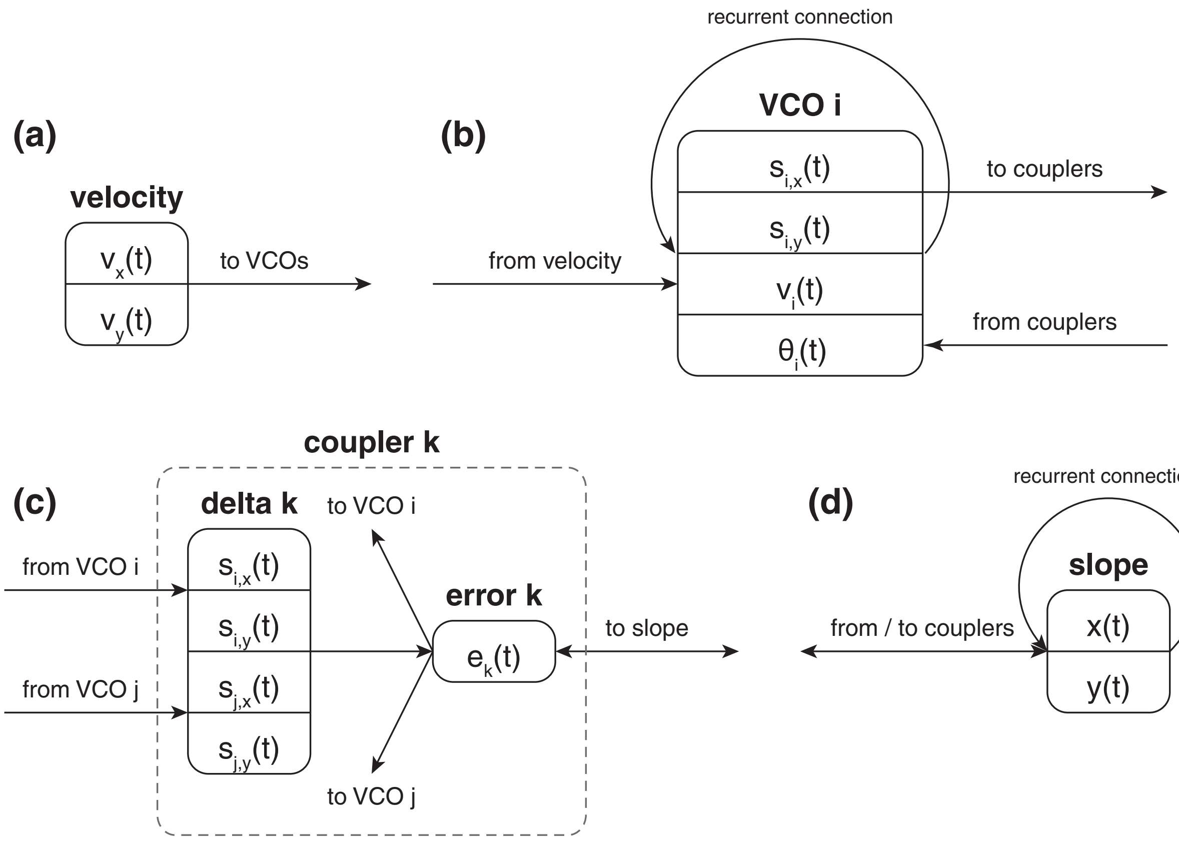

Figure 3.2: Network decomposed into parts. **(a)** a velocity node, **(b)** a VCO node, **(c)** a coupler node and **(d)** a slope node. The connections to/from the node and the values stored in the node are displayed.

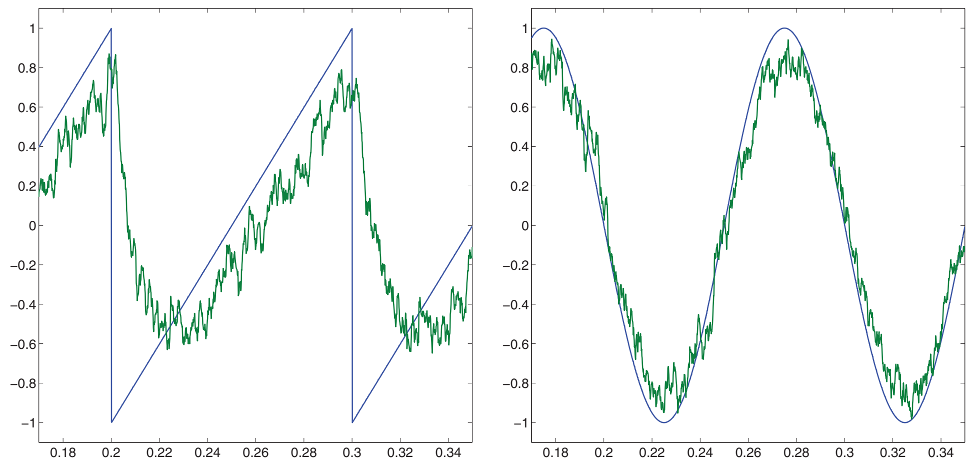
Figure 3.3: Discontinuity distortion in neural modeling. **Left.** Neural simulation (green) of a 10 Hz sawtooth wave (blue). Severe distortion happens when the value jumps from 1 to -1. **Right.** Neural simulation (green) of a 10 Hz sine wave (blue).

3.3.3 VCO Node

The VCO nodes embody the system's core function of encoding the animal's position into phases. In addition, they receive inputs from couplers to combat drift.

A VCO has four dimensions, so that VCO i stores $s_{i,x}(t)$, $s_{i,y}(t)$, $v_i(t)$ and $\theta_i(t)$, as shown in Figure 3.2 (b).

The first two elements $(s_{i,x}(t), s_{i,y}(t))$ serve as the phase vector. Here we choose to encode the phase vector because other representations do not perform as well in neural network modeling. Encoding the phase angle is not possible, since it increases infinitely (because of $\phi_b(t)$), and neural ensembles have a limited coding range. It is also inadequate to encode $\phi_i(t) \bmod 2\pi$, because encoding the jump-discontinuity from 0 to 2π is problematic using neuron ensembles. Figure 3.3 illustrates the difficulties of decoding this discontinuity. In contrast, $s_{i,x}(t)$ and $s_{i,y}(t)$ are bounded and smooth, which makes them ideal variables for neural encoding.

The third dimension $v_i(t)$ is the modulated velocity, as shown in Equation 3.21. In essence, $v_i(t)$ is the frequency adjustment (from baseline $\omega(b)$) induced by the animal's motion. Movement in the direction (c_x, c_y) will increase the VCO's frequency, while movement in the opposite direction will decrease it.

The last element $\theta_i(t)$ is the adjustment sent from the couplers, which will be discussed in the next section.

Apart from connections from the velocity node and coupler nodes, VCO nodes have recurrent connections that result in their oscillating function. Representing the phase with $\phi_i(t)$, we wish the recurrent connection to implement the phase update

$$\begin{aligned}\phi_i(t+\delta t) &= \phi_i(t) + \omega_i(t)\delta t + \theta_i(t) \\ &= \phi_i(t) + \Big(v_i(t) + \omega_b(t)\Big)\delta t + \theta_i(t)\ ,\end{aligned} \tag{3.23}$$

which basically adds the coupling adjustment to Equation 2.3. We use δt to represent a very short time period. As a normal theta-wave frequency [37] , $\omega_b(t)$ is set at 8 Hz. Noticing that $\phi_i(t+\delta t) - \phi_i(t)$ is relatively small (typically between -0.1 and 0.1 radians), we can update the phase vector with a first-order approximation, namely

$$\begin{aligned}s_{i,x}(t+\delta t) &= \cos\phi_i(t+\delta t) \\ &\approx \cos\phi_i(t) + \Big(\big(v_i(t)+\omega_b(t)\big)\delta t + \theta_i(t)\Big)\cos'\phi_i(t) \\ &= \cos\phi_i(t) - \Big(\big(v_i(t)+\omega_b(t)\big)\delta t + \theta_i(t)\Big)\sin\phi_i(t) \\ &= s_{i,x}(t) - \Big(\big(v_i(t)+\omega_b(t)\big)\delta t + \theta_i(t)\Big)s_{i,y}(t)\ ,\end{aligned} \tag{3.24}$$

$$\begin{aligned}s_{i,y}(t+\delta t) &= \sin\phi_i(t+\delta t) \\ &\approx \sin\phi_i(t) + \Big(\big(v_i(t)+\omega_b(t)\big)\delta t + \theta_i(t)\Big)\sin'\phi_i(t) \\ &= \sin\phi_i(t) + \Big(\big(v_i(t)+\omega_b(t)\big)\delta t + \theta_i(t)\Big)\cos\phi_i(t) \\ &= s_{i,y}(t) + \Big(\big(v_i(t)+\omega_b(t)\big)\delta t + \theta_i(t)\Big)s_{i,x}(t)\ .\end{aligned} \tag{3.25}$$

Figure 3.4 illustrates this approximation. This system is essentially the Euler's method solution for the simple harmonic oscillator.

After updating the phase vector as shown above, we divide both elements in the vector by the vector's magnitude, in order to prevent the oscillating radius from over expanding or shrinking.

The VCO nodes pass both components of their phase vectors as input to coupler nodes.

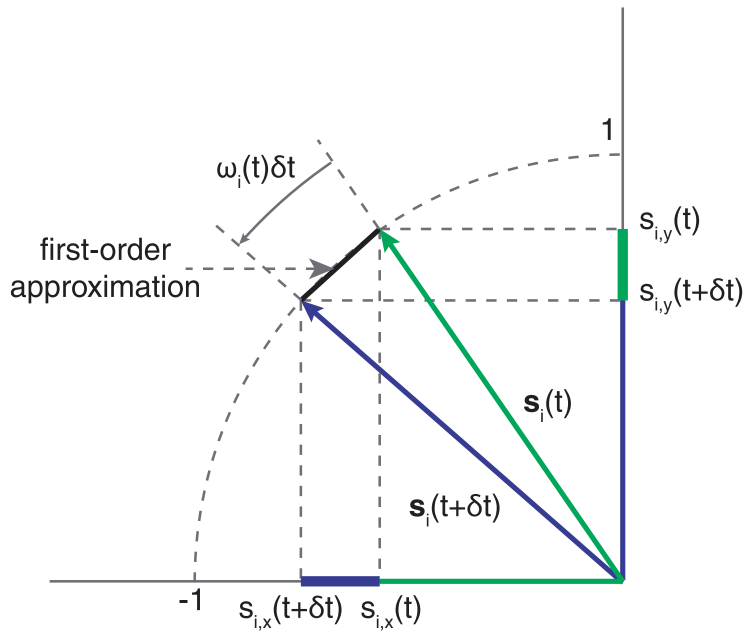

Figure 3.4: A first-order approximation of phase vector update. The incremental phase angle is approximated by the black linear segment, following Equation 3.25.

3.3.4 Coupler Node

Each coupler node connects two VCOs, computes their phase difference and sends back the phase adjustment.

A coupler node can be divided into two parts: a *delta node* and an *error node*, as shown in Figure 3.2 (c). The 4-dimensional delta node assembles the phase vectors from the two VCOs. For example, if coupler node k connects VCO nodes i_k and j_k ($k \in \{1, 2, \ldots, m\}$, $i_k, j_k \in \{1, 2, \ldots, n\}$, $i_k \neq j_k$), the four dimensions of delta node k will be $s_{i_k,x}(t)$, $s_{i_k,y}(t)$, $s_{j_k,x}(t)$ and $s_{j_k,y}(t)$.

Delta node k computes the phase difference $\Delta\phi_k(t)$ between VCO nodes i_k and j_k, and sends it to error node k. We can derive the complex form of $\Delta\phi_k(t)$ by

$$\begin{aligned}
& \exp\Big(i\Delta\phi_k(t)\Big) \\
= \; & \exp\Big(i\big(\phi_{i_k}(t) - \phi_{j_k}(t)\big)\Big) \\
= \; & \exp\Big(i\phi_{i_k}(t)\Big)\exp\Big(-i\phi_{j_k}(t)\Big) \\
= \; & \Big(\cos\phi_{i_k}(t) + i\sin\phi_{i_k}(t)\Big)\Big(\cos\phi_{j_k}(t) - i\sin\phi_{j_k}(t)\Big) \\
= \; & \Big(s_{i_k,x}(t)s_{j_k,x}(t) + s_{i_k,y}(t)s_{j_k,y}(t)\Big) + i\Big(s_{i_k,y}(t)s_{j_k,x}(t) - s_{i_k,x}(t)s_{j_k,y}(t)\Big) \; . \qquad (3.26)
\end{aligned}$$

When the phase difference is small, we can approximate $\Delta\phi_k(t)$ by the imaginary part of

$$\exp\Big(i\Delta\phi_k(t)\Big) \; ,$$

so that

$$\Delta\phi_k(t) \approx s_{i_k,y}(t)s_{j_k,x}(t) - s_{i_k,x}(t)s_{j_k,y}(t) \; . \qquad (3.27)$$

Error nodes store and update $\boldsymbol{e}(t)$ as described in Equation 3.18,

$$\boldsymbol{e}(t) = \Delta\boldsymbol{\phi}(t) - \Delta\boldsymbol{C}\boldsymbol{p}'(t) \; ,$$

which, when expanded, yields

$$\begin{bmatrix} e_1(t) \\ e_2(t) \\ \vdots \\ e_m(t) \end{bmatrix} = \begin{bmatrix} \Delta\phi_1(t) \\ \Delta\phi_2(t) \\ \vdots \\ \Delta\phi_m(t) \end{bmatrix} - \begin{bmatrix} \Delta c_{1,x} & \Delta c_{1,y} \\ \Delta c_{2,x} & \Delta c_{2,y} \\ \vdots & \vdots \\ \Delta c_{m,x} & \Delta c_{m,y} \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \; . \qquad (3.28)$$

Each error node encodes a single component in $\boldsymbol{e}(t)$, so error node k stores

$$e_k(t) = \Delta\phi_k(t) - \Big(\Delta c_{k,x}x(t) + \Delta c_{k,y}y(t)\Big) \ , \tag{3.29}$$

in which

$$\begin{aligned}\Delta c_{k,x} &= c_{i_k,x} - c_{j_k,x} \\ \Delta c_{k,y} &= c_{i_k,y} - c_{j_k,y} \ ,\end{aligned}$$

as shown in Equation 3.8. The phase difference $\Delta\phi_k(t)$ comes from delta node k as mentioned above, and the other part $\Delta c_{k,x}x(t) + \Delta c_{k,y}y(t)$ comes from the slope node, which will be discussed in the next section.

The output of the error nodes provide updates to both VCO nodes and the slope node, following rules given by Equations 3.19 and 3.20, which can also be written as

$$\begin{aligned}\Delta\boldsymbol{\phi}(t+\delta t) &= \Delta\boldsymbol{\phi}(t) - \gamma\boldsymbol{e}(t) \\ \boldsymbol{p}'(t+\delta t) &= \boldsymbol{p}'(t) + \gamma\Delta\boldsymbol{C}^{\mathrm{T}}\boldsymbol{e}(t) \ .\end{aligned}$$

Error node k corrects $\Delta\phi_k(t)$ by sending feedback to its corresponding VCOs, adjusting their phase vectors to get closer to the desired phase difference. It sends half of the correction to each VCO, so that

$$\frac{-\gamma e_k(t)}{2}$$

goes to $\theta_{i_k}(t)$, and

$$\frac{\gamma e_k(t)}{2}$$

goes to $\theta_{j_k}(t)$. Hence, $\phi_{i_k}(t) - \phi_{j_k}(t)$ is reduced by $\gamma e_k(t)$.

Regarding the correction to the slope node, error node k sends a 2-dimensional vector,

$$\delta\boldsymbol{p}'_k(t) = \gamma e_k(t)\begin{bmatrix}\Delta c_{k,x} \\ \Delta c_{k,y}\end{bmatrix} \ , \tag{3.30}$$

to the slope node. Since all error nodes are connected to the slope node, the input from error nodes to the slope node is the sum of $\delta\boldsymbol{p}'_k(t)$ for $k \in \{1, 2, \ldots, m\}$, which gives

$$\delta\boldsymbol{p}'(t) = \sum_{k=1}^{m} \gamma e_k(t)\begin{bmatrix}\Delta c_{k,x} \\ \Delta c_{k,y}\end{bmatrix} = \gamma\Delta\boldsymbol{C}^{\mathrm{T}}\boldsymbol{e}(t) \ , \tag{3.31}$$

matching the adjustment described in Equation 3.20. We set γ to 0.5, which enables fast enough convergence while preventing the gradient descent process from oscillating.

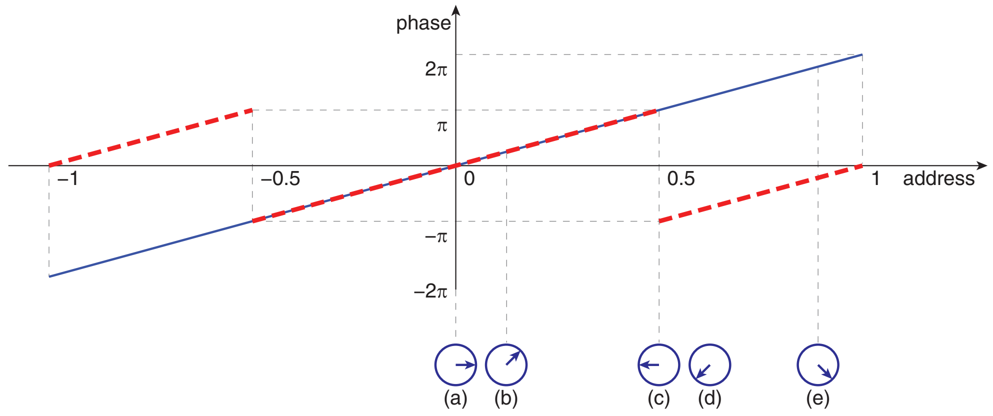

Figure 3.5: Demonstration of phase wrapping. VCOs are arranged in 1-D, and their phases form a 1-D ramp (blue). Due to their periodicity, the phase vectors can also be translated into "wrapped" phases (red). Five example VCOs are shown at **(a)** 0, **(b)** 0.125, **(c)** 0.5, **(d)** 0.625 and **(e)** 0.875.

Coupling scheme. Given that we have n VCOs that can be coupled two at a time, we are faced with the decision of which pairs of VCOs to couple. Phase vectors can only represent angles in a 2π range. Thus, we cannot represent phase differences greater than 2π. The further apart two VCOs, the bigger their phase difference for a given phase ramp. Hence, coupling VCOs that are far apart is only useful if the slope of the phase ramp is small (thus avoiding phase differences larger than 2π). Coupling nearby VCOs gives us a wider range of ramp slopes, while still avoiding the phase-wrapping problem.

Consider a 1-D phase ramp as shown in blue in Figure 3.5. With an address radius of 1, the phases reach a maximum of 2π and a minimum of -2π. Suppose we have five VCOs, positioned as shown in the figure. We are able to couple VCOs (a) and (b), or VCOs (c) and (d), because they are close enough so that we can compute the phase differences only by looking at their phase vectors. But for VCOs (a) and (c), for example, from their phase vectors we cannot determine whether their phase difference should be $-\pi$ or π. Another example is VCO pair (a) and (e), since the difference in phase vector indicates a small negative angle, but the actual phase difference is almost 2π.

This restriction affects our design in two directions. To ensure that only VCOs with a phase difference less than 2π are connected by a coupler, we can either limit the phase ramp slope, or limit the distance between coupled VCOs. We explore both limitations in our

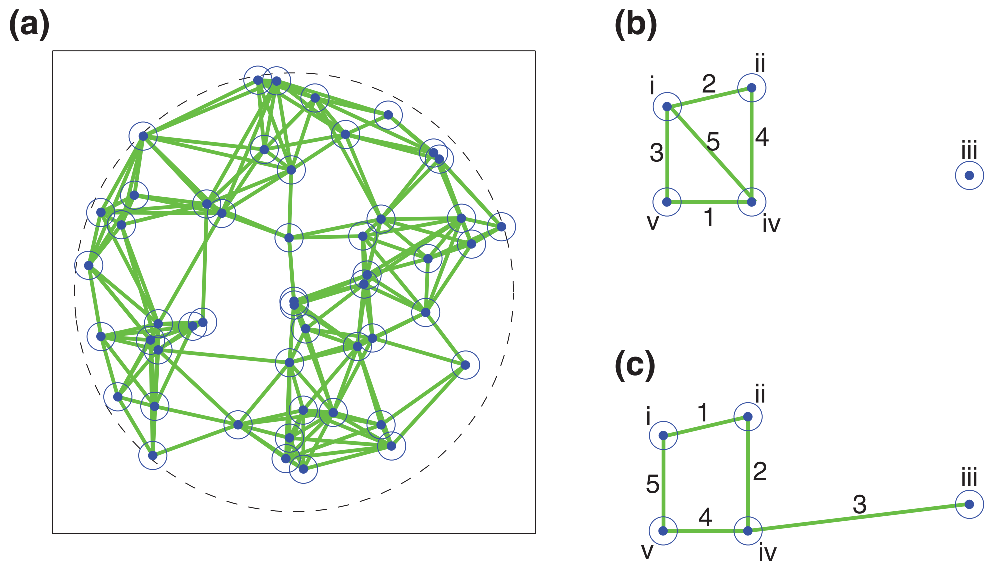

Figure 3.6: Local coupling. **(a)** Demonstration of local coupling with 50 randomly placed VCOs following a uniform distribution (blue) and 200 couplers (green). **(b)** Minimum distance coupling (MDC) with five VCOs (blue) and five couplers (green). Couplers are added one by one, ordered by number 1 to 5. The new coupler always connects the closest uncoupled VCO pair. **(c)** Connected minimum distance coupling (CMDC) with five VCOs (blue) and five couplers (green). Couplers are added one by one, ordered by number 1 to 5. The scheme traverses VCOs ordered by i, ii, iii, iv and v. The new coupler connects the current VCO to its closed uncoupled VCO.

system. For the latter one, we implement it by always preferring to couple nearby VCOs than to couple distant ones, an approach named *local coupling*. Local coupling avoids phase wrapping by only coupling VCOs that are close to each other (Figure 3.6 (a)).

We raise two different local coupling schemes. The first one, called *Minimum Distance Coupling* (MDC), looks at all uncoupled pairs, and couples the two VCOs that are closest to each other. Hence, this method achieves the minimum global coupling distance (Figure 3.6 (b)). The other coupling scheme is called *Connected Minimum Distance Coupling* (CMDC). It visits the VCOs in order, coupling each to the nearest VCO that it is not already coupled to (Figure 3.6 (c)). While sacrificing some locality, CMDC tries to spread couplers evenly among VCOs. This is different from the MDC, since the MDC scheme makes it possible to have VCOs not coupled to anything.

To study whether local coupling affects the system's behavior, we can also limit the phase ramp slope to an extent that some long-range couplings can be added into the system. We restrict the animal's movement to a unit circle, so that the magnitude of the phase ramp slope cannot exceed 1. With VCO addresses also in a unit circle, the largest phase difference between any two VCOs should be less than 2. In this way, we are able to make couplings between any two VCOs.

Chapters 4 and 5 give more details about how different coupling schemes and long-range couplings will affect the system's behavior.

3.3.5 Slope Node

The slope node maintains the current estimate of the phase ramp slope, accepting corrections sent from all error nodes. It contains two dimensions

$$\boldsymbol{p}'(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$$

as shown in Equation 3.12, which, as mentioned before, are numerically equivalent to the coordinates of the animal's location. The slope node is also recurrently connected to itself, so that it behaves like a leaky integrator. Working together with the input from error nodes, the slope is updated following Equation 3.20, namely

$$\boldsymbol{p}'(t + \delta t) = \boldsymbol{p}'(t) + \gamma \Delta \boldsymbol{C}^{\mathrm{T}} \boldsymbol{e}(t) \ .$$

The slope node sends output to all error nodes to help compute $\boldsymbol{e}(t)$. Error node k receives the input from the slope node as

$$-\Delta c_{k,x} x(t) - \Delta c_{k,y} y(t) \ ,$$

so that the total input to the error node becomes

$$\Delta \phi_k(t) - \Big(\Delta c_{k,x} x(t) + \Delta c_{k,y} y(t) \Big) \ ,$$

as described in Equation 3.29.

A detailed view of the network architecture is shown in Figure 3.7.

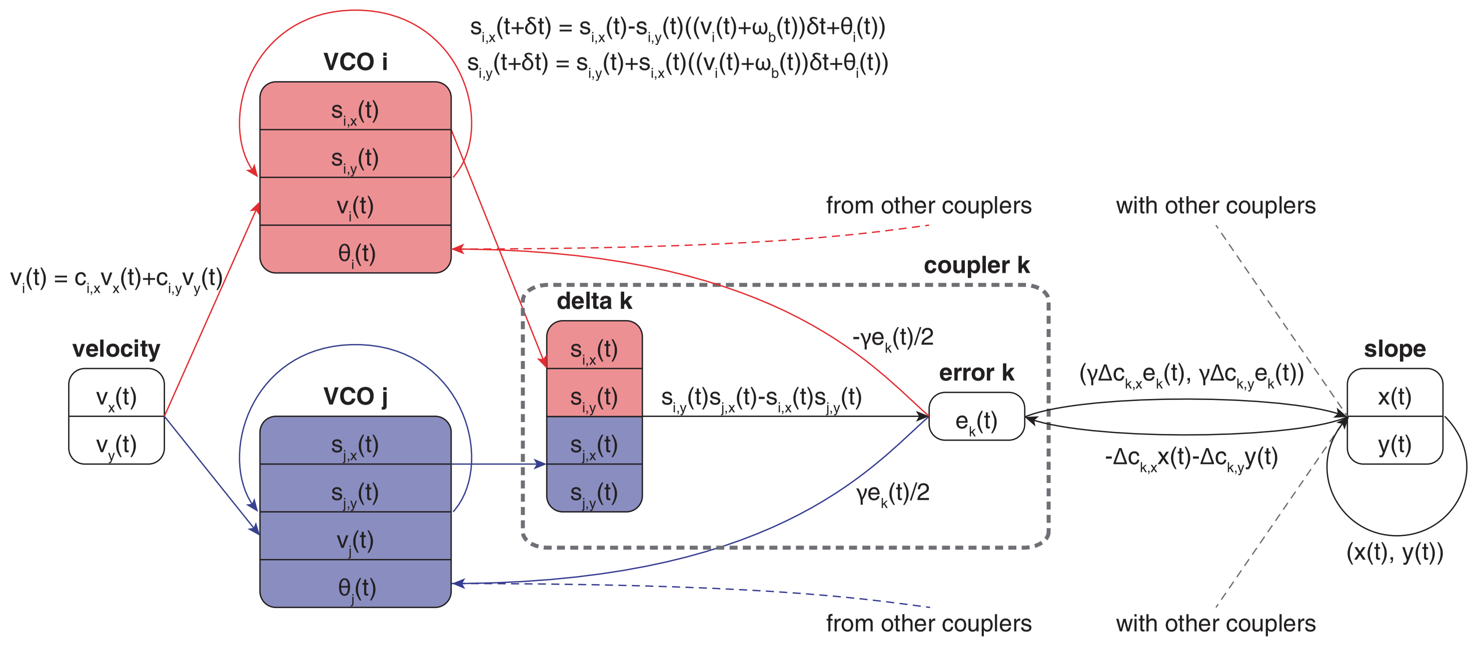
$s_{i,x}(t+\delta t) = s_{i,x}(t)-s_{i,y}(t)((v_i(t)+\omega_b(t))\delta t+\theta_i(t))$
$s_{i,y}(t+\delta t) = s_{i,y}(t)+s_{i,x}(t)((v_i(t)+\omega_b(t))\delta t+\theta_i(t))$
VCO i
$s_{i,x}(t)$
$s_{i,y}(t)$
$v_i(t)$
$\theta_i(t)$
$v_i(t) = c_{i,x}v_x(t)+c_{i,y}v_y(t)$
velocity
$v_x(t)$
$v_y(t)$
VCO j
$s_{j,x}(t)$
$s_{j,y}(t)$
$v_j(t)$
$\theta_j(t)$
from other couplers
with other couplers
coupler k
delta k
$s_{i,x}(t)$
$s_{i,y}(t)$
$s_{j,x}(t)$
$s_{j,y}(t)$
$-\gamma e_k(t)/2$
error k
$s_{i,y}(t)s_{j,x}(t)-s_{i,x}(t)s_{j,y}(t)$
$e_k(t)$
$\gamma e_k(t)/2$
$(\gamma\Delta c_{k,x}e_k(t), \gamma\Delta c_{k,y}e_k(t))$
$-\Delta c_{k,x}x(t)-\Delta c_{k,y}y(t)$
slope
$x(t)$
$y(t)$
$(x(t), y(t))$
from other couplers
with other couplers

Figure 3.7: Details of an example network, composed of a velocity node, two VCO nodes, a coupler node and a slope node. VCO i (red) and VCO j (blue) are connected to coupler k. Data stored and values passed are displayed.

Chapter 4

Experiments

A series of experiments are conducted to characterize the performance of this generalized framework. The purpose is twofold: one is to test the framework's behavior with different arguments, and the other is to compare the performance of the new network with that of the 3-propeller model [28].

4.1 Environment

All experiments were run on a PC with Intel i5-4670 processor, 16GB RAM, and Windows 8.1 64-bit operating system. They all used Nengo 1.4 for neuron modeling and neural network simulation. Data analysis and graph plotting was on the same machine and operating system with Matlab R2014a 64-bit.

4.2 Quality Measures

We propose two quantitative quality measures to evaluate the models' performance.

Reconstruction error. *Reconstruction error* is defined as the Euclidean distance between the animal's perceived position, reconstructed from the VCO phase ramp, and the animal's actual position. More specifically, reconstruction error at time t, denoted as $E_r(t)$, can be represented as

$$E_r(t) = \sqrt{\Big(x_r(t) - x(t)\Big)^2 + \Big(y_r(t) - y(t)\Big)^2}\ , \tag{4.1}$$

in which $x_r(t)$ and $y_r(t)$ are the coordinates of the estimated (perceived) position stored in the plane node, and $x(t)$ and $y(t)$ are the coordinates of the actual position. Since it is essential for a path integration system to encode positions accurately, smaller $E_r(t)$ reflects better performance of a model.

Phase variance. As we mentioned above, VCOs tend to drift out of phase because of the inherent noise of neural oscillators. Notice that even for a fixed phase ramp slope, there is still a degree of freedom for the VCO phases: their variance. They could tightly adhere to the ramp, or could be dispersed more widely, deviating above and below the ramp. With the same reconstruction error, the further away VCO phases are from the ramp, the bigger the variance is, which makes the system less stable. Thus, it is important to measure to what extent VCO phases adhere to (or deviate from) the phase ramp. We define *phase variance* at time t, denoted as $E_c(t)$, as

$$E_c(t) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Big(c_{i,x}x_r(t) + c_{i,y}y_r(t) + \phi_b(t) - \phi_i(t)\Big)^2}\ , \tag{4.2}$$

in which $c_{i,x}x(t)$, $c_{i,y}y(t)$, $\phi_b(t)$ and $\phi_i(t)$ are related as mentioned in Equation 2.6, namely

$$\phi_i(t) = c_{i,x}x(t) + c_{i,y}y(t) + \phi_b(t)\ .$$

Notice that Equation 2.6 only stands in ideal situations, where $E_c(t)$ must be zero. In fact, Equation 4.2 exactly measures the disagreement between the left and right of Equation 2.6 caused by neural noise. In other words, $E_c(t)$ is the root mean square of the difference between actual VCO phases and what they should be according to the fitted ramp. We also use the term *coplanarity*, defined as the reciprocal of phase variance. Smaller $E_c(t)$ indicates better coplanarity and more robustness of the system.

4.3 Model Parameters

Several parameters may affect the network's performance.

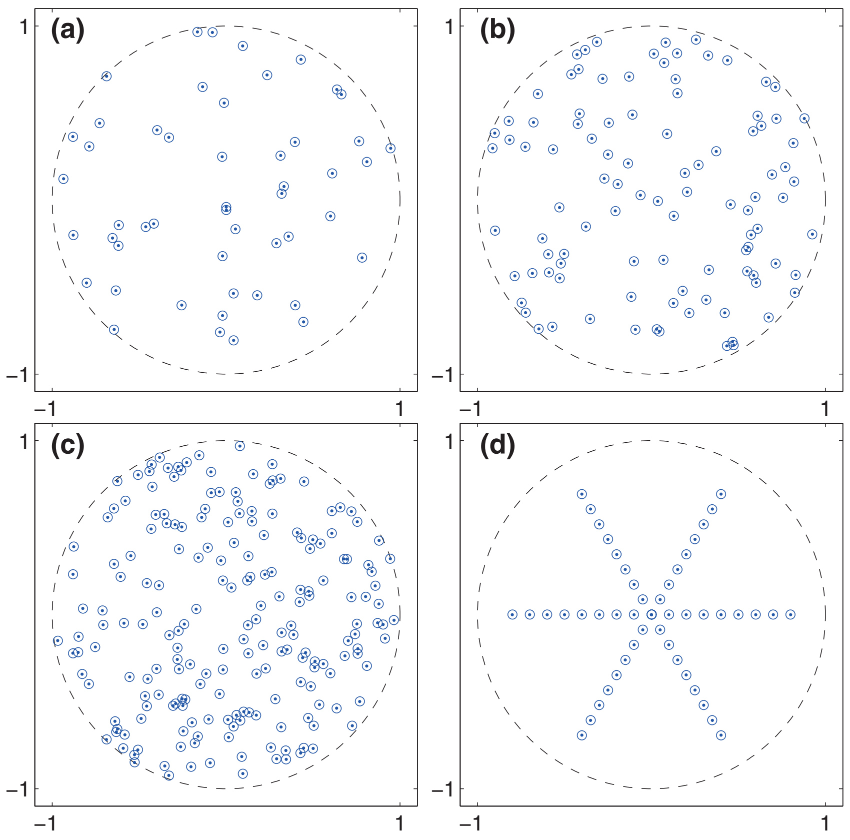

Figure 4.1: VCO arrangement in different models. Plots show VCOs (blue) in the 2-D address space with **(a)** 50, **(b)** 100 and **(c)** 200 randomly placed VCOs with a uniform distribution. **(d)** The VCO arrangement of the 3-propeller model in the address space. Dashed circles are centered at $(0, 0)$ and have a radius of 1, marking the boundary of VCO addresses.

VCO count. We define the VCO count as the number of VCOs we use in our network. We tested networks with 50, 100 and 200 VCOs, all randomly placed in the address space inside the unit circle following a uniform distribution. How the VCOs are exactly arranged can be seen in Figure 4.1.

Coupling density. We define coupling density as the number of couplers divided by the number of VCOs. We tested 4 different coupling densities: 1, 2, 3, and 4. For example, when using 100 VCOs, we ran experiments with 100, 200, 300, and 400 couplers. A demonstration of different coupling densities with 50 VCOs is shown in Figure 4.2 (a).

Long-range coupling. We also include a modified condition when coupling density is equal to 1, substituting 10% of the couplers with long-range couplers. These couplers are set up in a way that each of them connects two groups of VCOs that did not have any couplers between them. Figure 4.2 (b) shows a comparison between models with and without long-range coupling.

Coupling scheme. Both Minimum Distance Coupling (MDC) and Connected Minimum Distance Coupling (CMDC) schemes are used in the experiments, in conjunction with different VCO counts and coupling densities. Examples of the two schemes are shown in Figure 4.2 (c).

We tested combinations of different VCO counts, coupling densities, and coupling schemes. All test cases are listed in Table 4.1, together with number of neurons used in each.

Table 4.1: Test cases

VCO amount	Coupling scheme	Coupler amount	Coupling density	Neurons used
50	MDC	50	1	46000
50	MDC	45 + 5 long-range	1	48500
50	MDC	100	2	71000
50	MDC	150	3	96000
50	MDC	200	4	121000
50	CMDC	50	1	46000
50	CMDC	45 + 5 long-range	1	48500
50	CMDC	100	2	71000
50	CMDC	150	3	96000
50	CMDC	200	4	121000

Continued on next page

Table 4.1 – *Continued from previous page*

VCO amount	Coupling scheme	Coupler amount	Coupling density	Neurons used
100	MDC	100	1	91000
100	MDC	90 + 10 long-range	1	96000
100	MDC	200	2	141000
100	MDC	300	3	191000
100	MDC	400	4	241000
100	CMDC	100	1	91000
100	CMDC	90 + 10 long-range	1	96000
100	CMDC	200	2	141000
100	CMDC	300	3	191000
100	CMDC	400	4	241000
200	MDC	200	1	181000
200	MDC	180 + 20 long-range	1	191000
200	MDC	400	2	281000
200	MDC	600	3	381000
200	MDC	800	4	481000
200	CMDC	200	1	181000
200	CMDC	180 + 20 long-range	1	191000
200	CMDC	400	2	281000
200	CMDC	600	3	381000
200	CMDC	800	4	481000
51	Propeller	48	0.94	63300

We ran 10 trials for each test case. Each trial is a simulation of the subject moving for 5 seconds, following a randomly generated track. The same 10 tracks are used across test cases for the purpose of comparison (Figure 4.3). The tracks are limited to the unit circle, imitating the Morris water maze as mentioned in Section 1.3. The simulated speed when following the track varies with time, with an average of 0.3 units per second.

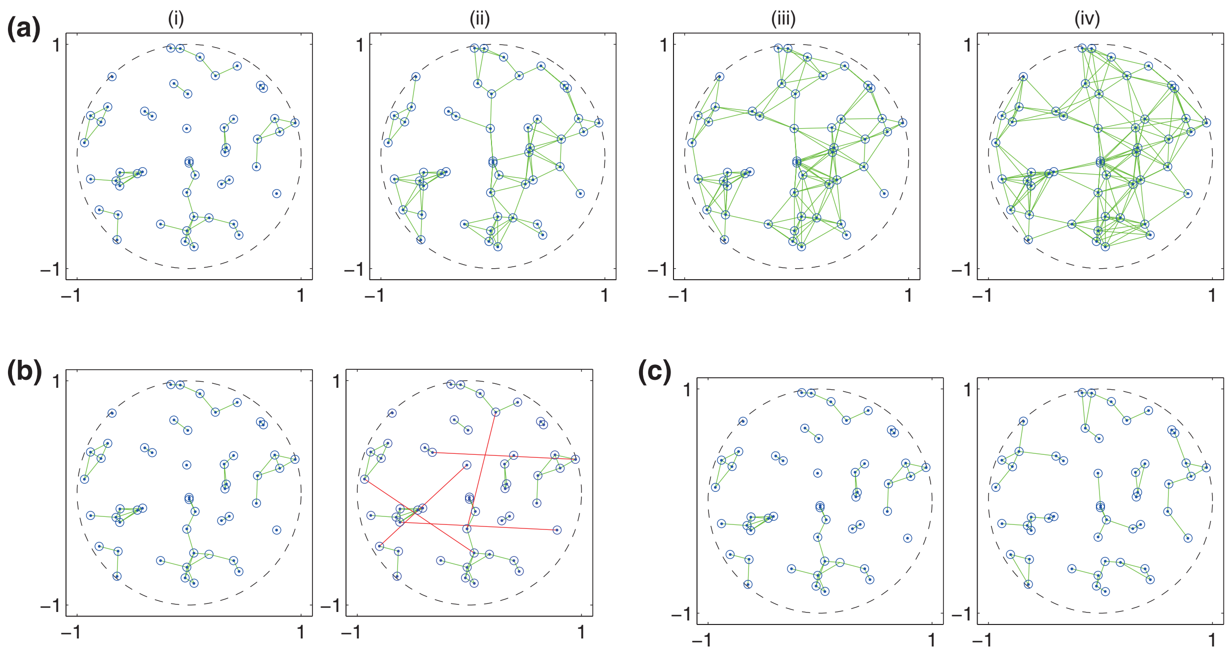

Figure 4.2: Coupling schemes and densities in different models. All plots show VCOs (blue circles with dots) and couplers (green or red lines) in the 2-D address space. **(a)** Coupling densities of 1, 2, 3 and 4 with 50 VCOs are shown in order from left to right. **(b)** Comparison between models with and without long-range coupling. Green lines indicate normal couplers, while red lines correspond to long-range couplers. **(c)** Comparison between MDC (left) and CMDC (right) with 50 VCOs and coupling density 1.

The reconstruction error $E_r(t)$ and phase variance $E_c(t)$ are recorded once per millisecond for every trial. The first 1 second is discarded because the system is not in a stable state. For every test case we can get $4 * 1000 * 10 = 40000$ data points. The mean and variance of $E_r(t)$ and $E_c(t)$ are computed from those 40000 points.

In addition, we connect a readout node to the VCOs, simulating a place cell. We ran experiments on a 120-second random track with 50, 100 and 200 VCOs, and a coupling density of 2. The spiking activity of the readout node was also recorded.

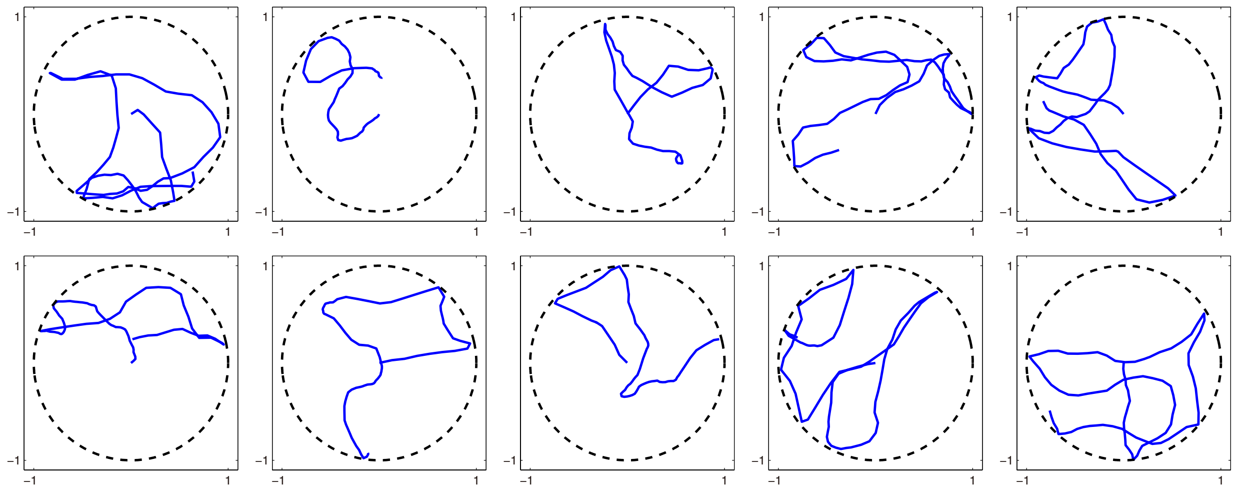

Figure 4.3: The 10 simulated tracks (blue) used in experiments. All tracks start at $(0, 0)$, last 5 seconds, and are limited in the unit circle.

4.4 Hypotheses

We raise several hypotheses about the relationship between parameters and model performance measures.

VCO count. Seeing VCOs as samples in the address space, and hence samples in the Fourier spectrum, we expect that increasing the number of VCOs provides more data for fitting, decreasing the reconstruction error. We hypothesize that phase variance is unrelated to VCO count.

Coupling density. We argue that denser coupling decreases the deviation of VCO phases from the fitted ramp, providing better coplanarity. Hence, we expect phase variance to decrease with increasing coupling density. Reconstruction error will also be reduced due to better robustness.

Long-range coupling. Targeted to increase the global connectivity, we believe that adding long-range couplers will result in better coplanarity (lower phase variance) and smaller reconstruction error.

Coupling scheme. Noticing that MDC makes more local couplings than CMDC, which leads to decreased global connectivity, we expect higher reconstruction error and higher phase variance when using MDC.

Reconstruction error versus phase variance. Since better coplanarity makes the system more robust, it is likely that the two quality measures are positively correlated.

Tests of these hypotheses are in the following chapter.

Chapter 5

Observations

The reconstruction error and phase variance for each test case is listed in Table 5.1. We supply this table for reference, but all the data is displayed in graph form in this chapter. All results are averaged over 10 trials with 4000 data points in each trial.

To make statistically meaningful comparisons between a pair of random variables, we look for their means to be significantly different from each other. We ran Wilcoxon signed-rank tests on all pairs of trials, and all the means were different at a statistical significance level of $p = 0.05$.

Table 5.1: Results

VCO amount	Coupling scheme	Coupler amount	Reconstruction error	Phase variance
50	MDC	50	0.428	0.385
50	MDC	45 + 5 long-range	0.204	0.183
50	MDC	100	0.112	0.152
50	MDC	150	0.083	0.113
50	MDC	200	0.080	0.097
50	CMDC	50	0.242	0.308
50	CMDC	45 + 5 long-range	0.161	0.206

Continued on next page

Table 5.1 – *Continued from previous page*

VCO amount	Coupling scheme	Coupler amount	Reconstruction error	Phase variance
50	CMDC	100	0.089	0.141
50	CMDC	150	0.079	0.116
50	CMDC	200	0.075	0.105
100	MDC	100	0.374	0.373
100	MDC	90 + 10 long-range	0.170	0.170
100	MDC	200	0.151	0.142
100	MDC	300	0.062	0.117
100	MDC	400	0.070	0.075
100	CMDC	100	0.246	0.299
100	CMDC	90 + 10 long-range	0.154	0.187
100	CMDC	200	0.079	0.184
100	CMDC	300	0.060	0.132
100	CMDC	400	0.081	0.094
200	MDC	200	0.490	0.375
200	MDC	180 + 20 long-range	0.213	0.163
200	MDC	400	0.101	0.127
200	MDC	600	0.049	0.109
200	MDC	800	0.057	0.068
200	CMDC	200	0.202	0.274
200	CMDC	180 + 20 long-range	0.119	0.161
200	CMDC	400	0.062	0.171
200	CMDC	600	0.058	0.115
200	CMDC	800	0.048	0.093
51	Propeller	48	0.361	0.194

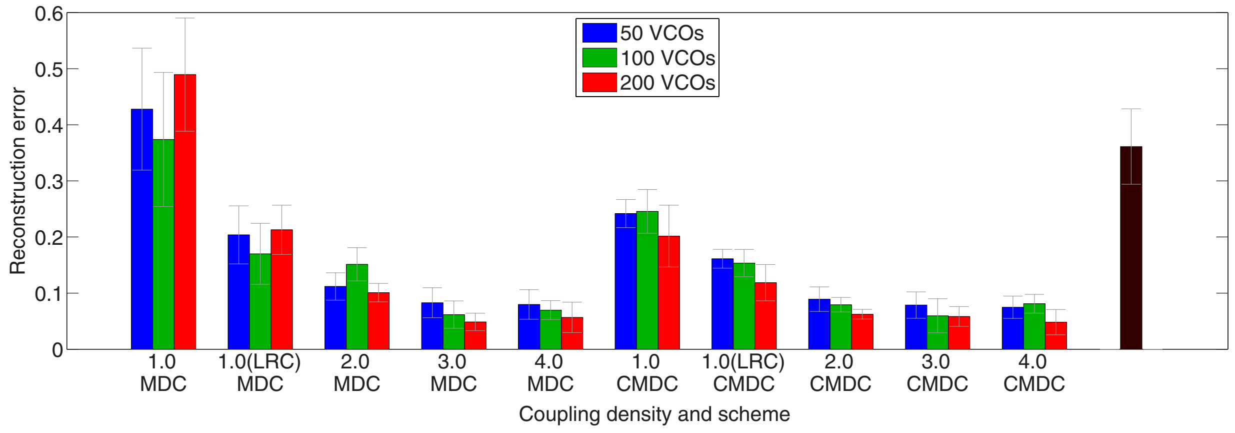

Figure 5.1: Reconstruction error vs. VCO count. Each bar corresponds to the reconstruction error averaged over ten 4-second trials in each test case. Bars are grouped by coupling densities and schemes. The rightmost bar shows the mean reconstruction error of the 3-propeller model. Grey error bars indicate one standard deviation. “LRC” stands for long-range coupling.

5.1 VCO Count

Reconstruction error is plotted in Figure 5.1 for a variety of different VCO counts. The corresponding phase variance is shown in Figure 5.2. Cases with the same coupling density and scheme are grouped together for comparison.

We cannot observe any obvious relationship between quality measures and VCO count. Although in many cases the reconstruction error and phase variance decrease when VCO count increases, other cases do not follow this pattern.

The amount of difference between cases is also limited. Among cases with the same coupling density and scheme, the maximal difference of reconstruction error is 23.6% of the larger one, while typically the difference lies around 15%. For phase variance, the difference is maximally 23.4%, and the majority of cases are between 5% and 15%.

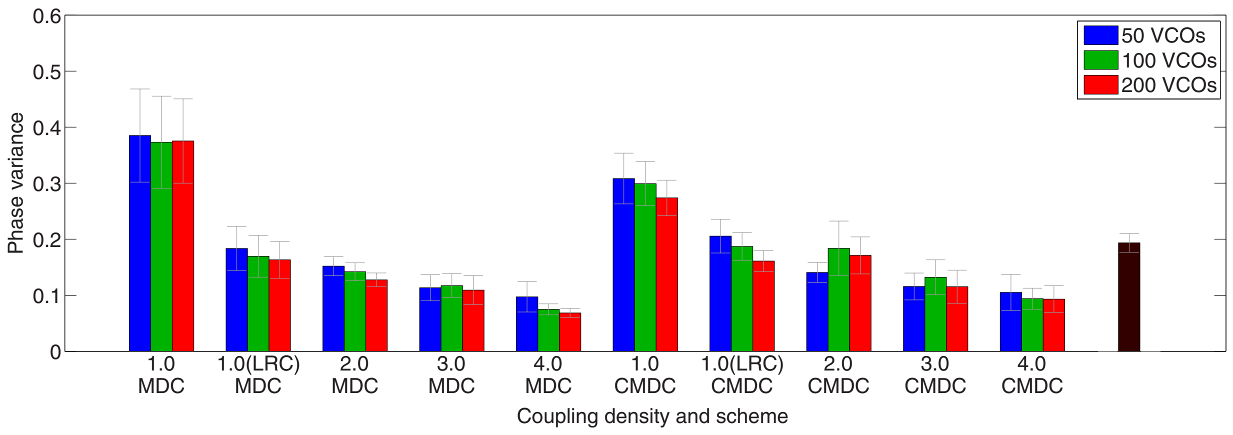

Figure 5.2: Phase variance vs. VCO count. Each bar corresponds to the phase variance averaged over ten 4-second trials in each test case. Bars are grouped by coupling densities and schemes. The rightmost bar shows the mean phase variance of the 3-propeller model. Grey error bars indicate one standard deviation. “LRC” stands for long-range coupling.

5.2 Coupling Density

We can see substantial trends when reconstruction error (Figure 5.3) and phase variance (Figure 5.4) are plotted over various coupling densities. Cases with the same VCO count and coupling scheme are grouped together for comparison.

Increasing coupling density from 1 to 2 greatly reduces reconstruction error. As shown in Figure 5.3, the reconstruction error for coupling density of 2 (red) is typically only about 30% to 40% of that for coupling density of 1 (blue). Increasing coupling density from 2 to 3 (cyan) also results in lower reconstruction error in all cases. The effect is comparatively weak, normally around 15%. There is no obvious trend in reconstruction error when increasing coupling density from 3 to 4 (purple).

Phase variance exhibits a similar pattern (Figure 5.4). It is greatly reduced when coupling density goes from 1 to 2, normally by 40% to 60%. Increasing coupling density from 2 to 3 and from 3 to 4 each yield further decreases in phase variance.

This observation broadly matches our expectation, which argues that denser coupling suppresses noise more effectively, bringing both less reconstruction error and better coplanarity. The fact that reconstruction error stops deceasing when coupling density increases

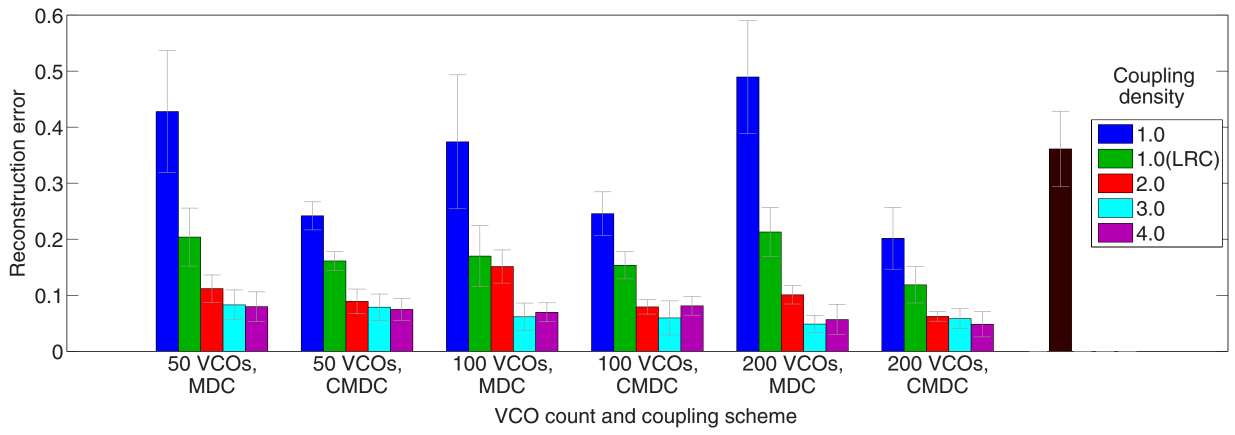

Figure 5.3: Reconstruction error vs. coupling density. Each bar corresponds to the reconstruction error averaged over ten 4-second trials in each test case. Bars are grouped by VCO counts and coupling schemes. The rightmost bar shows the mean reconstruction error of the 3-propeller model. Grey error bars indicate one standard deviation. "LRC" stands for long-range coupling.

from 3 to 4 is possibly because when coupling density grows beyond 3, the coupling is so strong that little noise is left for further inhibition.

5.3 Long-range Coupling

Reconstruction error and phase variance with long-range coupling are indicated in Figure 5.3 and Figure 5.4 by green bars. We observe that using long-range coupling improves models' performance. With the same coupling density of 1, reconstruction error for cases with long-range coupling is reduced to only about 50% to 70% of those for cases without long-range coupling. A similar pattern exists for phase variance. With long-range coupling, phase variance decreases by 40% to 60%. Models with coupling density of 1 and long-range coupling give similar phase variance to those with a coupling density of 2. The results imply that, apart from increasing coupling density, making VCOs more globally coupled is another effective approach to enhance coupling strength.

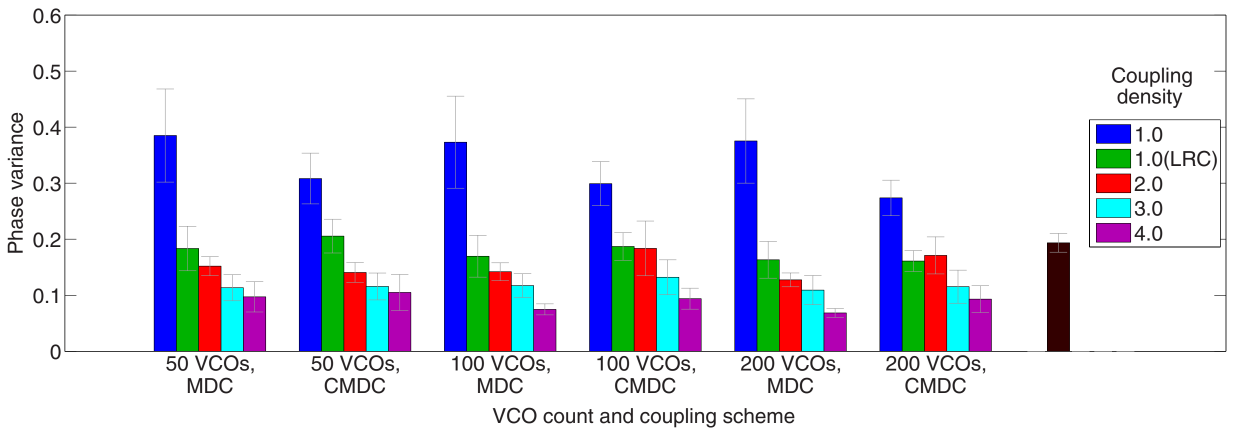

Figure 5.4: Phase variance vs. coupling density. Each bar corresponds to the phase variance averaged over ten 4-second trials in each test case. Bars are grouped by VCO counts and coupling schemes. The rightmost bar shows the mean phase variance of the 3-propeller model. Grey error bars indicate one standard deviation. "LRC" stands for long-range coupling.

5.4 Coupling Scheme

Regarding reconstruction error (Figure 5.5), the difference between MDC and CMDC is obvious with a coupling density of 1 (group 1). Reconstruction error with CMDC tends to be only 40% to 60% of that with MDC. The trend is not distinguishable when long-range coupling exists or coupling density goes beyond 2.

A similar situation happens with phase variance. Phase variance with CMDC is smaller with a coupling density of 1 (group 1), usually around 80% of that with MDC. There is also no obvious trend in other cases.

We infer from the results that increasing global connectivity reduces both reconstruction error and phase variance. We observe that clusters of nearby VCOs are often coupled to form locally-connected subgraphs. When coupling density is low, while VCOs in each subgraph may be well coupled, VCOs in different subgraphs can drift out of phase since there is no connection between them. This may account for the fact that CMDC gives better results, since CMDC tends to connect VCOs more globally than MDC. However, this effect is only obvious when coupling density is small and without long-range coupling (which increases global connectivity). As Figure 4.2 (a) shows, VCOs are connected globally in

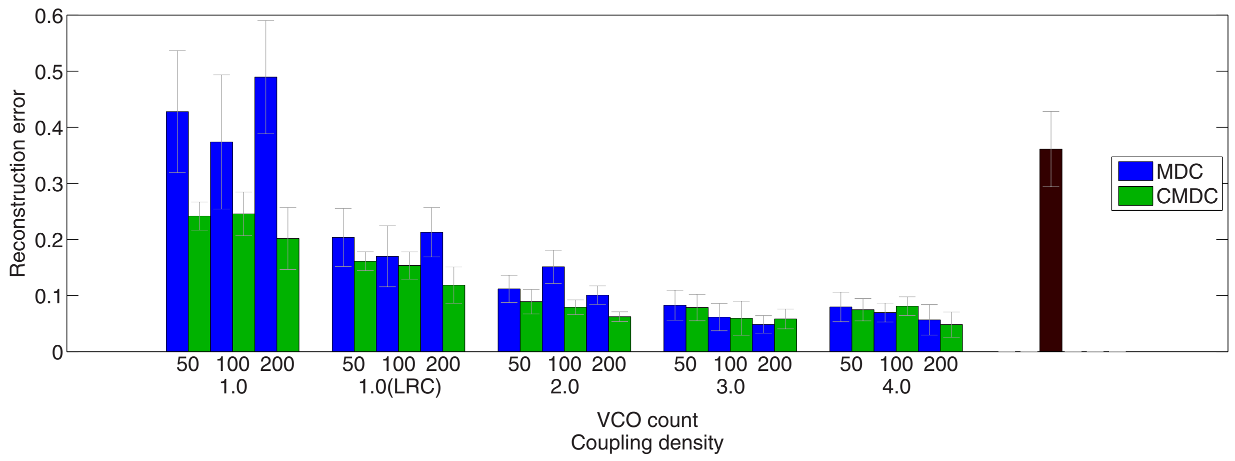

Figure 5.5: Reconstruction error vs. coupling scheme. Each bar corresponds to the reconstruction error averaged over 10 4-second trials in each test case. Bars are grouped by coupling densities. The rightmost bar shows the mean reconstruction error of the 3-propeller model. Grey error bars indicate one standard deviation. “LRC” stands for long-range coupling.

either scheme when coupling is dense enough. This can explain why there is no noticeable difference between MDC and CMDC with long-range couplers or higher coupling density.

5.5 Comparison with the 3-propeller Model

We compare the performance of our previous model with that of models under the new framework. Figure 5.7 and Figure 5.8 show the reconstruction error and the phase variance of different test cases sorted in descending order.

Results show that three models exhibit larger reconstruction error than our previous model, all with MDC scheme and a coupling density of 1. Regarding phase variance, all models with a coupling density of 1 behave worse than the 3-propeller model, while all other models show better performance except the one with 50 VCOs, CMDC scheme, and a coupling density of 1 with long-range coupling. The results suggest that, with certain parameters, our new network is capable of giving better performance than our previous model. We should notice, though, that not all networks used the same number of neurons.

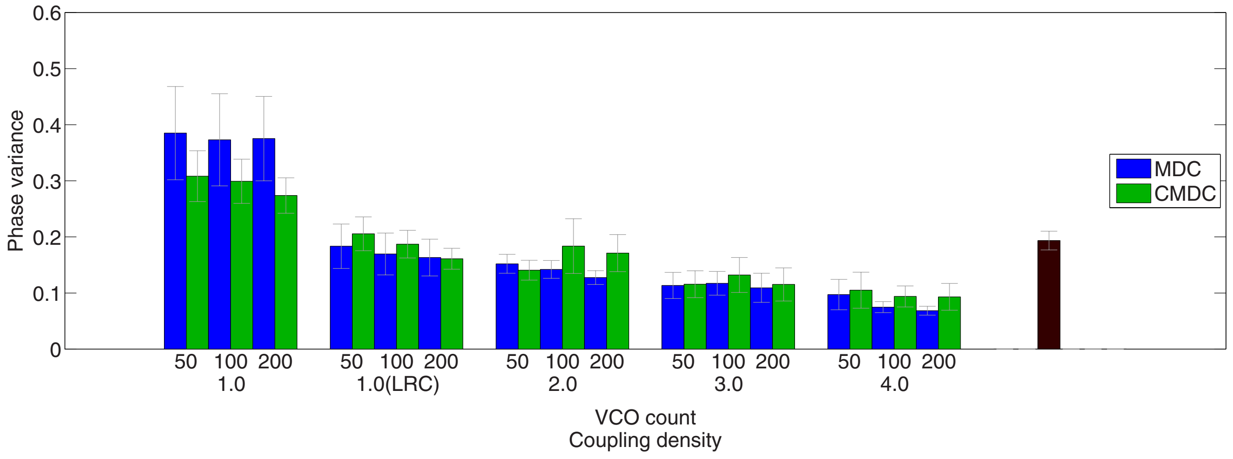

Figure 5.6: Phase variance vs. coupling scheme. Each bar corresponds to the phase variance averaged over 10 4-second trials in each test case. Bars are grouped by coupling densities. The rightmost bar shows the mean phase variance of the 3-propeller model. Grey error bars indicate one standard deviation. "LRC" stands for long-range coupling.

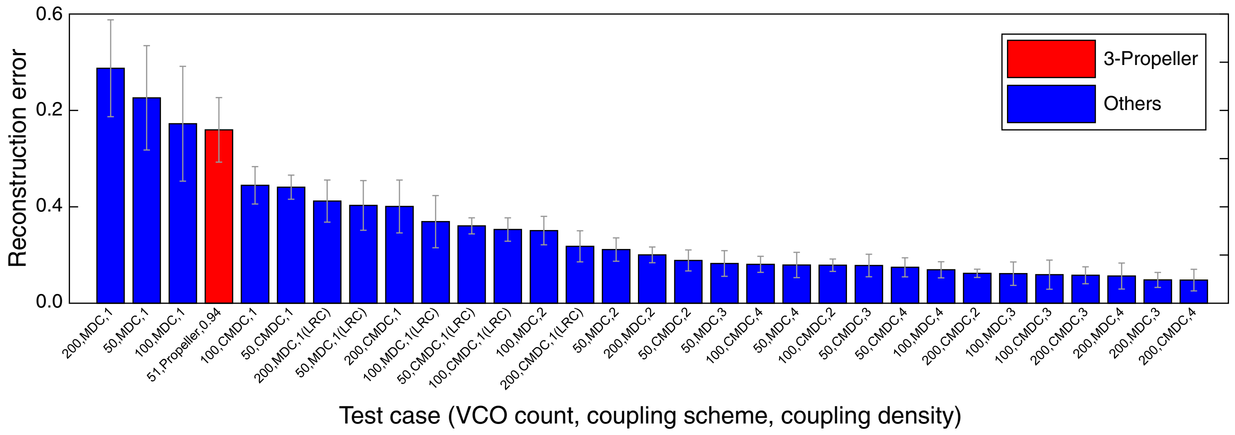

Figure 5.7: Reconstruction error sorted in descending order. Each bar corresponds to the reconstruction error averaged over 10 4-second trials in each test case. The red bar shows the reconstruction error of the 3-propeller model, while blue bars correspond to other cases. Grey error bars indicate one standard deviation. "LRC" stands for long-range coupling.

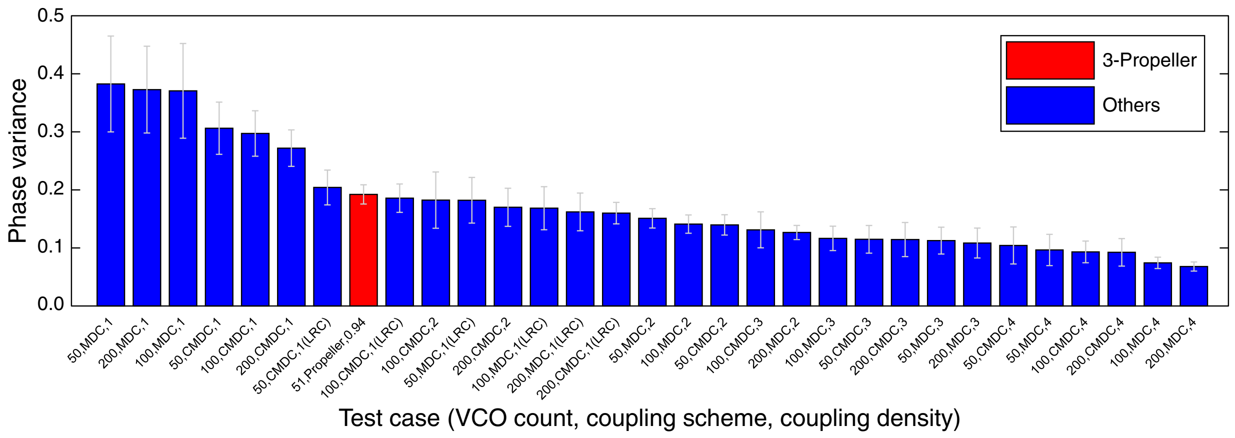

Figure 5.8: Phase variance sorted in descending order. Each bar corresponds to the phase variance averaged over 10 4-second trials in each test case. The red bar shows the phase variance of the 3-propeller model, while blue bars correspond to other cases. Grey error bars indicate one standard deviation. “LRC” stands for long-range coupling.

5.6 Reconstruction Error Versus Phase Variance

In Figure 5.9 we see that reconstruction error and phase variance are positively correlated, with $R^2 = 0.79$. This is within our expectation, as phase variance is negatively related with the stability of the estimated phase ramp, so that high phase variance correlates with high phase ramp slope variance, which may result in larger reconstruction error.

5.7 Place Cell

Apart from the quantitative measures, we are also interested in whether the model functions well as a generator of place cells. As an example, suppose we want to create a place cell with a firing field at the location with coordinates $(0.3, 0.4)$, as shown in Figure 5.10 (a). We connect a readout node to the VCOs to generate place cells. Following Equation 2.15, namely

$$w(c_x, c_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a(x, y) e^{-2\pi i(\phi-\phi_b)}\, \mathrm{d}x\mathrm{d}y \ ,$$

we can determine the connection weights using the Fourier transform [28].

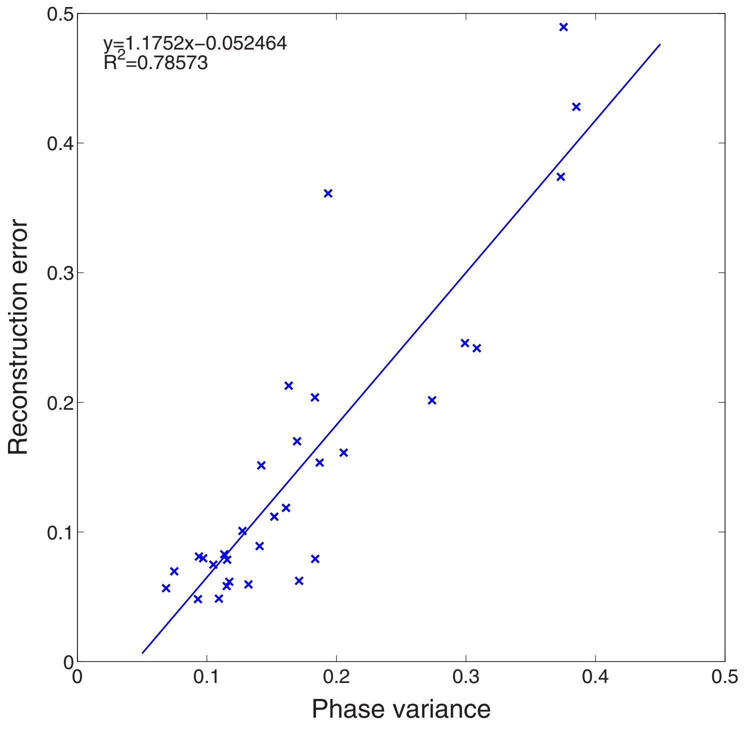

Figure 5.9: Reconstruction error vs. phase variance. Each mark corresponds to a test case, averaged over ten 4-second trials. The line gives the least-squares linear regression fit. The equation of the regression line and R^2 are shown at the top left.

We examined neurons in the readout node, and successfully found neurons that fire only when the animal is near $(0.3, 0.4)$, exhibiting the pattern of a place cell. Example place cells are shown in Figure 5.10. We also notice that increasing the VCO count generates a more centralized distribution of spikes, exactly as one would expect from a more densely-sampled Fourier spectrum.

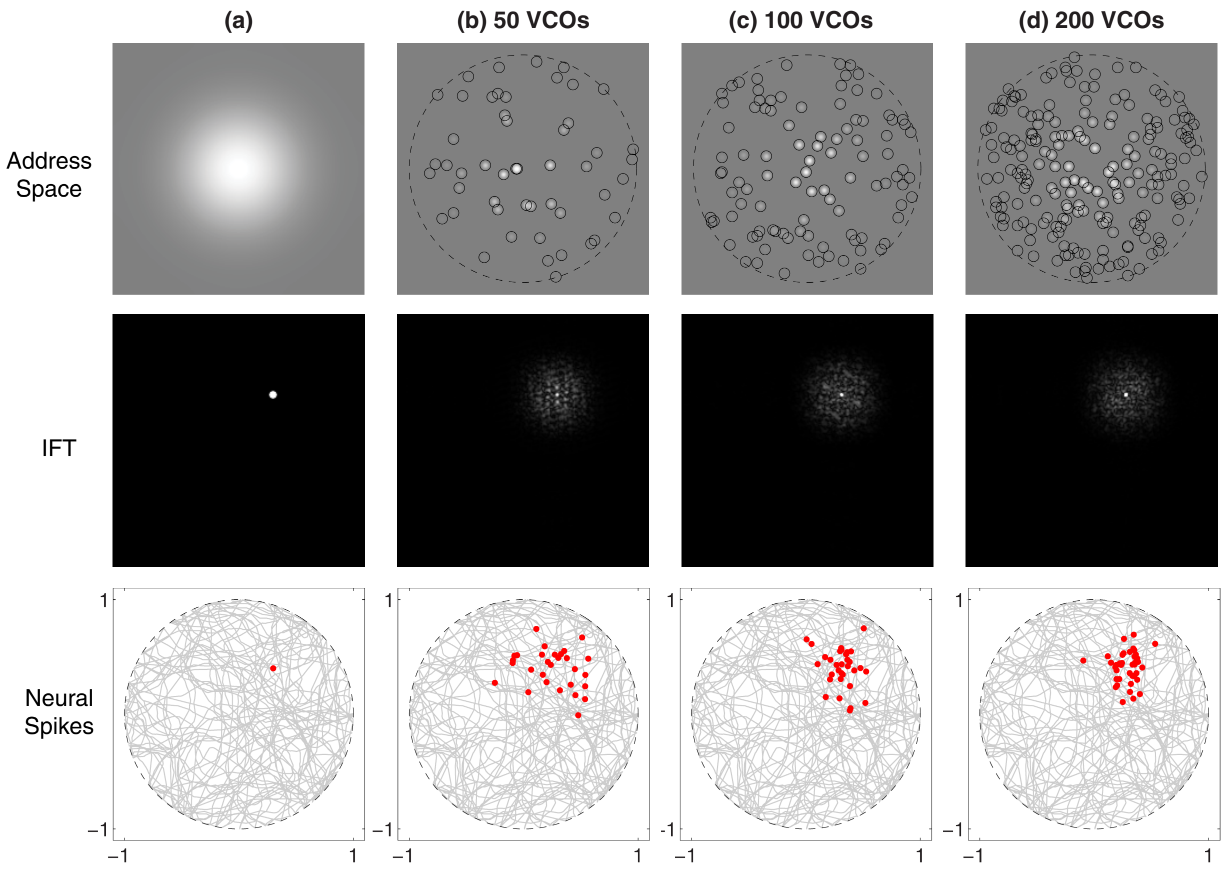

Figure 5.10: Place cells in different models. **(a)** Theory. The place shown by a white point (middle) corresponds to a set of Fourier coefficients in the frequency domain (top). The brightness shows the modulus of these weights, grey for 0 and white for 1. If we use the whole domain as the input of the readout node, in an ideal situation, the place cell should only fire at the exact place (bottom). The grey line shows the 120-second track, while the red dot gives the place where the cell fires. **(b)-(d)** Experiment results. The frequency domain is sampled by randomly distributed VCOs, indicated by black circles (top). Following an inverse Fourier transform, different VCO arrangements result in different spatial patterns, approximating the original firing field (middle). Experiments with these VCOs demonstrate that neurons firing in similar patterns exist (bottom).

Chapter 6

Conclusions

Our previous work argues that VCO phases form a linear ramp when VCOs are arranged in the address space. The slope of the phase ramp encodes the animal's position. We also suggest that we can obtain a neuron with an arbitrary firing field by connecting it to the VCOs with weights specified by Fourier theory. We constructed a prototype implementing the theory with restricted VCO arrangement. A rather contrived coupling scheme is involved to keep VCO phases from drifting.

In this thesis, we extend that previous work in various ways. First, we reform the theory to allow non-regular VCO arrangements. There exists a linear relationship between the VCO phases and the corresponding VCO address. The animal's position can be reconstructed as a least-squares solution of this linear relationship. VCO phase coupling can be achieved through iteratively minimizing the deviation of VCO phase differences from the estimated phase ramp slope.

Second, we implement a generalized path integration network using spiking LIF neurons. We incorporate randomly arranged VCOs, and employ local coupling schemes to avoid the ambiguity caused by the periodicity of phase vector oscillations.

Third, we test how parameter changes may affect the network's behavior. We propose two quantitative measures to evaluate the network's performance: reconstruction error, which measures how much the estimated (perceived) position deviates from the actual position, and phase variance, describing how much VCO phases deviate from the fitted phase ramp. We state various hypotheses about how VCO count, coupling density, long-range coupling and coupling scheme may affect the network's performance, evaluated by these two quality measures. The performance of our previous model is included for comparison. We also study the relationship between the two measures, as well as test the network's ability to serve as the input to place cells.

Our results show that some parameters have substantial influence on the network's behavior. Increasing coupling density greatly reduces reconstruction error and phase variance, especially when coupling density is small. Adding long-range coupling also notably improves the network's performance. Regarding coupling scheme, CMDC reduces reconstruction error and phase variance when coupling density is 1. The effect is slight with higher coupling density. On the other hand, VCO count does not seem to have an obvious influence on either quality measure. In addition, we observe that reconstruction error is positively related with phase variance. We also successfully generate a place cell receiving input from our path integration network. The data suggests that, although VCO count does not affect reconstruction error and coupling scheme, having more VCOs enables firing fields with higher spatial resolution. The results validate our generalized framework as a functional path integration system.

Some of the results above may shed light on how to implement a more effective path integration network.

One observation is that increasing coupling density greatly reduces reconstruction error and phase variance, but only up to a coupling density around 3. This may suggest an upper limit to effective coupling density. Moreover, considering a grid cell firing pattern can be generated by connecting to 3 VCOs, this coupling density limit might become some indirect evidence suggesting that grid cells themselves can be used as couplers. In fact, several recent models raised similar ideas about how grid cells may be used to stabilize VCO phases. Bush et al. [4] suggest that by connecting grid cells to form a continuous attractor network, grid cells are able to provide relative stability to VCO activities. Burgess et al. [2] gives a similar idea by connecting three evenly distributed VCOs to a grid cell. They show that the grid cell can be used to maintain relatively constant VCO phases. Likewise, the work by Blair et al. [1] suggests reciprocal connections between grid cells and some specific "theta cells" might control phase noise to correct the error in path integration. They also propose that a primary function of spatially tuned neurons might be to couple the phases of neural oscillators in a manner that allows them to encode spatial locations as patterns of neural synchrony.

Another observation is that CMDC behaves much better than MDC with low coupling density, which implies that the degree of global coupling has a substantial impact on the network's performance.

Apart from these possibilities, we suggest several extensions to our current work that may lead to future studies.

The first concern is about how properties of neurons may affect the system's behavior. By using the NEF we are able to focus on the high level structure of neural networks, but those "hidden" parameters used by the NEF to simulate neurons are still worth attention. For example, there exists a temporal delay when a signal is passed between neurons through synapses, which is modeled in the NEF by a *post synaptic time constant* (PSTC). In our implementation we chose biological realistic PSTC values (10 milliseconds) that enables the system to function properly, while it should be interesting to see if the behavior will change when the PSTC values follow a distribution that is similar to the distribution that exists in real neurons.

Another example is neural noise, which is implemented in our model in a way that each neuron possesses independent noise following the same distribution. However, in real neural networks neural noise comes from many sources [35], so that noise in each neuron contains both independent and dependent components, and it is possible that noise in different neurons does not follow the same distribution. Considering that the coupling mechanism in our work is designed to inhibit independent noise, it should be important to investigate if the coupling mechanism is still effective when the noise is following the model of the real neural noise.

The second task is to study how other VCO arrangements may affect the system's behavior. Our experiments only employ random VCO distributions, but we guess that certain VCO arrangements may either improve or worsen the network's performance.

Next is to have couplers that connect to more than two VCOs. Considering that the interference pattern of three VCOs may result in a grid cell [39], it is possible that these multi-way couplers themselves contain neurons with interesting firing fields.

The last extension is about long-range couplers. They seem to help with accuracy, but the fact that we can only encode phase differences modulo 2π limits their usage. There are, however, phase relationships for which the absolute phase drops out of the equation. For example, two VCOs arranged diametrically opposite each other (so that they are reflections of each other across the origin) are expected to have opposite phase offsets from the reference phase. Thus, their actual phase does not matter; we can assess their compliance simply by enforcing that their phase offsets be opposites. Similarly, our original model took advantage of the fact that adding the phase vectors of a triad of phase-step nodes (see Figure 2.2 (d)) should give a total phase angle of zero. With randomly-distributed VCOs, though, there is no guarantee that we will have such relationships. However, what if the VCOs could shift within the address space in an effort to optimize the phase coding? It seems possible that VCOs could adapt their velocity gains to reach an optimal solution for a path integration system.

APPENDICES

A Neural Engineering Framework

Our network is built on the Neural Engineering Framework (NEF) [6], a platform capable of large-scale neural network modeling [7]. In the NEF, we store values in neuron populations, where each population serves as a "node" in the neural network. A population is composed of a group of neurons with different responses to a certain input, so that their responses can be used as a group of basis functions to construct a variety of different output functions.

The NEF is comprised of three main principles: (1) population encoding and decoding, (2) neural transformations, and (3) neural dynamics.

A.1 Population Encoding and Decoding

In this framework, data is stored in the collective activities of a population of leaky integrate-and-fire (LIF) neurons. A population of N neurons (a node) can encode a vector $\mathbf{x}$ in its neural activities using

$$a_n(t) = G_n\left(\mathbf{x}(t) \cdot \mathbf{e}_n \alpha_n + \beta_n\right), \quad n \in \{1, 2, \cdots, N\} \tag{1}$$

where $\mathbf{e}_n$ is the encoding vector (preferred direction vector), and α_n and β_n are scalar gain and bias terms that account for the neural climate of the neuron. The input to the function $G_n(\cdot)$ can be thought of as the input current driving neuron n. The function G_n translates the input current to neural activity, either in the form of a firing rate, or a series of spikes. In the case of firing rate, G_n gives the steady-state firing rate, known as the tuning curve, mapping the input current to the neuron firing rate,

$$G_n(J) = \begin{cases} \frac{1}{\tau_{\mathrm{ref}} - \tau_{\mathrm{m}} \ln\left(1 - \frac{J_{\mathrm{th}}}{J}\right)} & \text{for } J > J_{\mathrm{th}} \\ 0 & \text{otherwise} \end{cases}$$

where τ_{ref} is the refractory period, τ_{m} is the membrane time constant, and J_{th} is the threshold current, below which the neuron has a firing rate of zero (Figure A1 (a)). On the other hand, if using spikes, then the output of G_n is represented as a sum of time-shifted Dirac delta functions [27],

$$G_n(J_n(t)) = \sum_p \delta(t - t_{np}) \ ,$$

where t_{np} is the time of the pth spike from neuron n. In this case, we model the membrane potential, v, using the differential equation,

$$\tau_m \frac{dv}{dt} = RJ_n(t) - v \ ,$$

where $J_n(t)$ is the input current, and R is the membrane resistance. Once the membrane potential reaches its threshold of v_{th} (which equals RJ_{th}), the neuron spikes, the timing of the spike is recorded, and the membrane potential is reset to zero.

If we wish to decode the neural activities of a population of neurons, we can compute the optimal linear decoders. We do this by collecting a sampling of inputs, X, and corresponding neural activities, A. That is, each row of matrix X stores a sample input, and each row of matrix A stores the corresponding neural activities for all N neurons (usually stored as firing rates). To decode from our population, we seek the linear weights D that solve

$$\min_D \|AD - X\|_2^2 \ .$$

Thus, the weights in D perform a linear transformation from the space of neural activities to the space of input values. This is a linear least-squares problem, and there are multiple ways to compute D numerically. Once we have D, we can decode neural activities to get an estimate of the value being encoded (Figure A1 (b)). Moreover, we can decode arbitrary functions of our encoded data by finding the decoders that solve

$$\min_D \|AD - f(X)\|_2^2 \ ,$$

where $f(X)$ is a function of the encoded values.

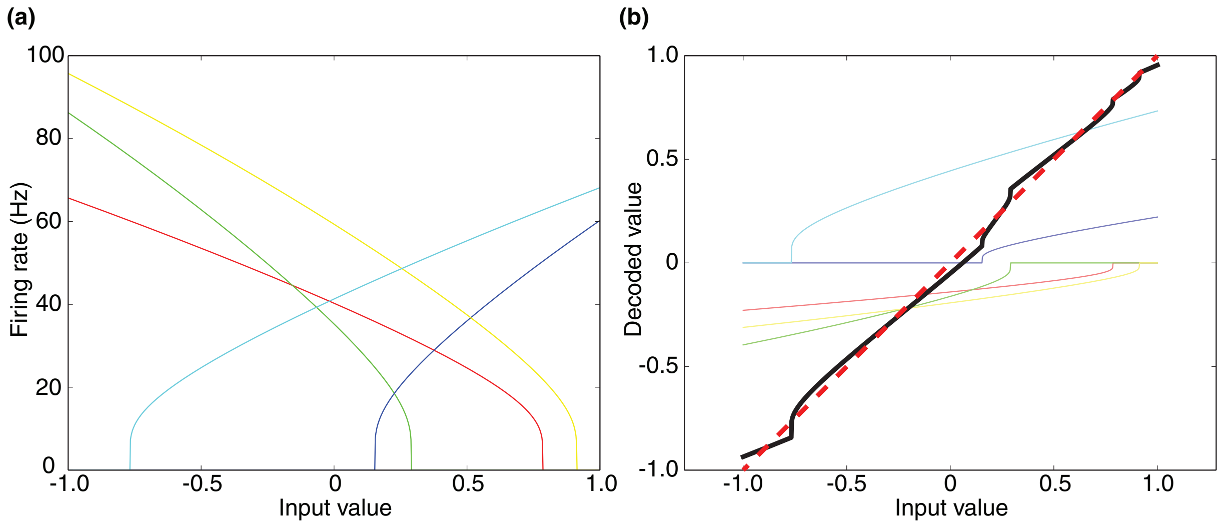

Figure A1: Population encoding and decoding in the NEF. **(a)** Tuning curves of five different neurons. The input value (x-axis) is encoded as firing rates (y-axis). **(b)** Decoding the input value from firing rates. Tuning curves are weighted so that adding the weighted firing rates gives a least-squares estimation (solid black line) of the encoded value. The ideal identity line is also shown (dashed red line).

A.2 Neural Transformations

We can use our neural encoders and decoders to transform data from one population P, to another population Q. To do this, we essentially decode the desired function from P and re-encode the result into Q. Collapsing those processes together gives the $N \times M$ weight matrix

$$W = D_P E_Q \alpha_Q \ ,$$

where D_P is the matrix that decodes the neural activities from the N neurons in P, and $E_Q \alpha_Q$ is the matrix of M encoders for the neurons in Q. The weight matrix simply combines the linear decoders and encoders into a single matrix.

A.3 Neural Dynamics

The neural coding and transformation principles above can be built into a dynamic framework by including the temporal action of synapses. When a spike arrives at a synapse, it induces a current on the post-synaptic neuron. We model this post-synaptic current using exponential decay[1]. That is, we convolve an incoming spike train with the post-synaptic filter, $h(t)$,

$$h(t) = \frac{1}{\tau_s} \exp\left(\frac{-t}{\tau_s}\right) \ , \tag{2}$$

where τ_s is the decay time constant for the synapse. The total current entering a neuron is a weighted sum over all incoming spike trains so that the total input current arriving at neuron m is

$$J_m(t) = h(t) \star \left[\sum_n w_{nm} \sum_p \delta(t - t_{np}) \right] \ ,$$

where $\star$ represents convolution, and ω_{nm} is the connection weight from neuron n to neuron m.

We can use a recurrently-connected population of neurons to implement a dynamic model of the form

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x})$$

[1]The NEF offers other post-synaptic filters.

by choosing the recurrent connection weights so that they decode and feed back $\tau_s f(\mathbf{x}) + \mathbf{x}$, where τ_s is the post-synaptic time constant [6].

With these three principles in place, we can implement a dynamical system using spiking LIF neurons by assigning populations to state variables, and connecting them with the appropriate transformations and time constants. A good example of such a dynamical system is the neural velocity-controlled oscillator, described in section 3.3.3.

B Experiment Python Code

```
import nef
import math
import random

tau = 0.005 #post-synaptic time constant
tau_osc = 0.01 #post-synaptic time constant for OSC
gamma = 0.5
status = 'default'

# test case settings
trial = 10
n_osc = 200
n_neib = 400
mode_neib = 'uniform'

# file paths
file_root = 'D:/Dropbox/random_coupler/Python/'
file_osc = file_root + 'osc/'
file_path = file_root + 'path/'
file_log = file_root + 'logs/' + '_'.join([str(n_osc), mode_neib, str(
    ↪ n_neib)]) + '/'

# get addresses
def rand_addresses(filename):
  f = open(file_osc + filename)
  res = []
  for line in f:
    numbers = [float(x) for x in line.split(',')]
    res.append(numbers)
  f.close()
  return res
addr = rand_addresses('addr_' + str(n_osc) + '.csv')

# get neighbors
```

```
def nearest_neighbors(filename):
  f = open(file_osc + filename)
  res = []
  for line in f:
    numbers = [int(x) for x in line.split(',')]
    res.append(numbers)
  f.close()
  return res
neib = nearest_neighbors('_'.join(['neib', str(n_osc), mode_neib, str(
    ↪ n_neib)]) + '.csv')

# get pairs
pairs = []
for i in range(n_osc):
  for j in range(n_osc):
    if neib[i][j] == 1:
      pairs.append([i, j])

# get counts
counts = [0 for x in range(n_osc)]
for p in pairs:
  counts[p[0]] += 1
  counts[p[1]] += 1

# get C
def get_C(addr, neib, nosc, nneib):
  res = [[0.0,0.0] for x in range(nneib)]
  index = 0
  for i in range(nosc):
    for j in range(nosc):
      if neib[i][j] == 1:
        res[index][0] = addr[i][0] - addr[j][0]
        res[index][1] = addr[i][1] - addr[j][1]
        index += 1
  return res
C = get_C(addr, neib, n_osc, n_neib)
```

```

# NETWORK SETUP

# network
net = nef.Network('Random addressed VCOs', seed=7)

# ensembles
osc = net.make_array('osc', neurons=400, length=n_osc, dimensions=4,
    ↪ radius=2, mode=status)
delta = net.make_array('delta', neurons=400, length=n_neib, dimensions
    ↪ =4, radius=1, mode=status)
error = net.make_array('error', neurons=100, length=n_neib, dimensions
    ↪ =1, radius=1, mode=status)
plane = net.make_array('plane', neurons=200, length=1, dimensions=2,
    ↪ radius=2, mode=status)

# input
class Rat(nef.SimpleNode):
  def init(self):
    # set trial string
    trial_string='path-140205-' + str(trial).zfill(2) + '.csv'
    f = open(file_path + trial_string, 'r')
    # read file
    self.data = []
    for line in f:
      values = [float(v) for v in line.split(',')]
      self.data.append(values)
    f.close()
    # set variables
    self.n = len(self.data)
    [self.vx, self.vy, self.x, self.y] = [0.0, 0.0, 0.0, 0.0]
  def tick(self):
    index = int(round(self.t * 1000))
    if index >= self.n:
      index = self.n - 1
    [self.vx, self.vy, self.x, self.y] = self.data[index]
  def origin_velocity(self):
    return [self.vx, self.vy]
  def origin_pos(self):
    return [self.x, self.y]
rat = Rat('Rat')
net.add(rat)

# VCO initialization
net.make_input('Impulse', [1], zero_after_time=0.005)

# logger
class Logger(nef.SimpleNode):

  def init(self):
    title_osc = [str(x) + '_' + str(y) for x in range(n_osc) for y in
       ↪ range(4)]
    f_osc = open(self.get_filename('osc'), 'w')
    f_osc.write(','.join(title_osc) + ',\n')
    f_osc.close()
    title_error = [str(x) + '_' + str(y) for x in range(n_neib) for y
       ↪ in range(2)]
    f_error = open(self.get_filename('error'), 'w')
    f_error.write(','.join(title_error) + ',\n')
    f_error.close()
    f_plane = open(self.get_filename('plane'), 'w')
    f_plane.write('x,y,\n')
    f_plane.close()

  def get_filename(self, name):
    return file_log + str(trial).zfill(2) + '_' + name + '.csv'

  def tick(self):
    f_osc = open(self.get_filename('osc'), 'a')
    for i in range(n_osc):
      o = osc.getNodes()[i].getOrigin('X').values.values
      f_osc.write(','.join([str(x) for x in o]) + ',')
    f_osc.write('\n')
    f_osc.close()
    f_error = open(self.get_filename('error'), 'a')
    for i in range(n_neib):
      a = error.getNodes()[i].getOrigin('X').values.values
      f_error.write(','.join([str(x) for x in a]) + ',')
    f_error.write('\n')
    f_error.close()
    f_plane = open(self.get_filename('plane'), 'a')
    p = plane.getNodes()[0].getOrigin('X').values.values
    f_plane.write(str(p[0]) + ',' + str(p[1]) + ',\n')
    f_plane.close()

logger = Logger('Logger')
net.add(logger)

# CONNECTION SETUP

# input-osc
input2osc = zeros((n_osc * 4, 2), typecode='f')
for i in range(n_osc):
  input2osc[i * 4 + 2][0] = addr[i][0]
  input2osc[i * 4 + 2][1] = addr[i][1]
net.connect(rat.getOrigin('velocity'), osc, transform=input2osc, pstc=
    ↪ tau)

# impulse-osc
impulse2osc = zeros((n_osc * 4, 1), typecode='f')
for i in range(n_osc):
  impulse2osc[i * 4] = 1
net.connect('Impulse', osc, transform=impulse2osc, pstc=tau)

# osc-osc
def osc2osc(x):
  dt = 1e-2
  freq = 10 + x[2]
  newx = x[0] - dt*x[1]*freq - x[1]*x[3] + 0.5 * (random.random() -
     ↪ 0.5)
  newy = x[1] + dt*x[0]*freq + x[0]*x[3] + 0.5 * (random.random() -
     ↪ 0.5)
  # Radius should be 1
  radius = sqrt( x[0]**2 + x[1]**2 )
  if radius > 1e-8:
    newx = newx / radius
    newy = newy / radius
  return newx, newy, 0.0, 0.0
net.connect(osc, osc, func=osc2osc, pstc=tau_osc)

# osc-delta
osc2delta = zeros((n_neib * 4, n_osc * 4), typecode='f')
for i in range(n_neib):
  o1 = pairs[i][0]
  o2 = pairs[i][1]
  osc2delta[i * 4 + 0][o1 * 4 + 0] = 1
  osc2delta[i * 4 + 1][o1 * 4 + 1] = 1
  osc2delta[i * 4 + 2][o2 * 4 + 0] = 1
  osc2delta[i * 4 + 3][o2 * 4 + 1] = 1
net.connect(osc, delta, transform=osc2delta, pstc=tau)

# delta-error
def delta2error(input):
  x1 = input[0]
  y1 = input[1]
  x2 = input[2]
  y2 = input[3]
  res = 1 * (x2 * y1 - x1 * y2)
  return res
net.connect(delta, error, func=delta2error, pstc=tau)

# error-plane
error2plane = zeros((2, n_neib * 1), typecode='f')
for i in range(n_neib):
  error2plane[0][i] = C[i][0] * gamma
  error2plane[1][i] = C[i][1] * gamma
net.connect(error, plane, transform=error2plane, pstc=tau)

# error-osc
error2osc = zeros((n_osc * 4, n_neib * 1), typecode='f')
for i in range(n_neib):
  o1 = pairs[i][0]
  o2 = pairs[i][1]
  error2osc[o1 * 4 + 3][i] = -0.4 / counts[o1]
  error2osc[o2 * 4 + 3][i] = 0.4 / counts[o2]
net.connect(error, osc, transform=error2osc, pstc=tau)

# plane-error
plane2error = zeros((n_neib * 1, 2), typecode='f')
for i in range(n_neib):
  plane2error[i][0] = -1 * C[i][0]
  plane2error[i][1] = -1 * C[i][1]
net.connect(plane, error, transform=plane2error, pstc=tau)

# plane-plane
plane2plane = zeros((2, 2), typecode='f')
plane2plane[0][0] = 1
plane2plane[1][1] = 1
net.connect(plane, plane, transform=plane2plane, pstc=tau)

# START

net.releaseMemory()
net.add_to_nengo()
net.run(5.1)
```

References

[1] Hugh T. Blair, Allan Wu, and Jason Cong. Oscillatory neurocomputing with ring attractors: a network architecture for mapping locations in space onto patterns of neural synchrony. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 369(1635):20120526, 2014.

[2] Christopher P Burgess and Neil Burgess. Controlling phase noise in oscillatory interference models of grid cell firing. *The Journal of Neuroscience*, 34(18):6224–6232, 2014.

[3] Neil Burgess, Caswell Barry, and John O'Keefe. An oscillatory interference model of grid cell firing. *Hippocampus*, 17(9):801–812, 2007.

[4] Daniel Bush and Neil Burgess. A hybrid oscillatory interference/continuous attractor network model of grid cell firing. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 34(14):5065–5079, 2014.

[5] John Conklin and Chris Eliasmith. A controlled attractor network model of path integration in the rat. *Journal of Computational Neuroscience*, 18(2):183–203, March 2005.

[6] Chris Eliasmith. *How to Build a Brain: A Neural Architecture for Biological Cognition*, volume 1. Oxford University Press, 1 edition, 2013.

[7] Chris Eliasmith, Terrence C Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. A large-scale model of the function brain. *Science*, 338(6111):1202–1205, 2012.

[8] Ariane S Etienne and Kathryn J Jeffery. Path integration in mammals. *Hippocampus*, 14(2):180192, 2004.

[9] Mark C Fuhs and David S Touretzky. A spin glass model of path integration in rat medial entorhinal cortex. *The Journal of Neuroscience*, 26(16):4266–4276, 2006.

[10] Caroline Geisler, David Robbe, Michael Zugaro, Anton Sirota, and Gyorgy Buzsaki. Hippocampal place cell assemblies are speed-controlled oscillators. *Proceedings of the National Academy of Sciences of the United States of America*, 104(19):8149–8154, 2007.

[11] Lisa M Giocomo, May-Britt B Moser, and Edvard I Moser. Computational models of grid cells. *Neuron*, 71(4):589–603, 2011.

[12] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt B Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

[13] Xiang Ji, Shrinu Kushagra, and Jeff Orchard. Sensory updates to combat path-integration drift. In Osmar R. Zaane and Sandra Zilles, editors, *Advances in Artificial Intelligence*, number 7884 in Lecture Notes in Computer Science, pages 263–270. Springer Berlin Heidelberg, 2013.

[14] James J Knierim, Joshua P Neunuebel, and Sachin S Deshmukh. Functional correlates of the lateral and medial entorhinal cortex: objects, path integration and local-global reference frames. *Philosophical Transactions of the Royal Society of London*, 369(1635):20130369, 2014.

[15] Christof Koch. Simplified models of individual neurons. In *Biophysics of Computation: Information Processing in Single Neurons*, volume 1. Oxford University Press, 1 edition, 1999.

[16] Julija Krupic, Neil Burgess, and John O'Keefe. Neural representations of location composed of spatially periodic bands. *Science*, 337(6096):853–857, 2012.

[17] Mate Lengyel, Zoltan Szatmary, and Peter Erdi. Dynamically detuned oscillations account for the coupled rate and temporal code of place cell firing. *Hippocampus*, 13(6):700–714, 2002.

[18] Eleanor A Maguire, David G Gadian, Ingrid S Johnsrude, Catriona D Good, John Ashburner, Richard SJ Frackowiak, and Christopher D Frith. Navigation-related structural change in the hippocampi of taxi drivers. *Proceedings of the National Academy of Sciences of the United States of America*, 97(8):4398–4403, 2000.

[19] Bruce L McNaughton, Francesco P Battaglia, Ole Jensen, Edvard I Moser, and May-Britt Moser. Path integration and the neural basis of the 'cognitive map'. *Nature Reviews Neuroscience*, 7(8):663–678, 2006.

[20] R G Morris, P Garrud, J N Rawlins, and J O'Keefe. Place navigation impaired in rats with hippocampal lesions. *Nature*, 297(5868):681–683, 1982.

[21] Robert U Muller, John L Kubie, E Bostock, J Taube, and G Quirk. Spatial firing correlates of neurons in the hippocampal formation of freely moving rats. *Brain and Space*, pages 296–333, 1991.

[22] Zaneta Navratilova and Bruce L Mcnaughton. Models of path integration in the hippocampal complex. In Dori Derdikman and James J Knierim, editors, *Space, Time and Memory in the Hippocampal Formation*, page 191224. Springer Vienna, Vienna, 2014.

[23] John O'Keefe. Place units in the hippocampus of the freely moving rat. *Experimental Neurology*, 51(1):78–109, 1976.

[24] John O'Keefe and Neil Burgess. Dual phase and rate coding in hippocampal place cells: theoretical significance and relationship to entorhinal grid cells. *Hippocampus*, 15(7):853–866, 2005.

[25] John O'keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*, volume 3. Clarendon Press Oxford, 1978.

[26] John O'Keefe and Michael L Recce. Phase relationship between hippocampal place units and the EEG theta rhythm. *Hippocampus*, 3(3):317–330, 1993.

[27] Alan V Oppenheim, Ronald W Schafer, John R Buck, and others. *Discrete-Time Signal Processing*. Prentice-hall Englewood Cliffs, 2 edition, 1989.

[28] Jeff Orchard, Hao Yang, and Xiang Ji. Does the entorhinal cortex use the fourier transform? *Frontiers in Computational Neuroscience*, 7:179, 2013.

[29] Jeff Orchard, Hao Yang, and Xiang Ji. Navigation by path integration and the fourier transform: A spiking-neuron model. In Osmar R. Zaane and Sandra Zilles, editors, *Advances in Artificial Intelligence*, number 7884 in Lecture Notes in Computer Science, pages 138–149. Springer Berlin Heidelberg, 2013.

[30] Gregory J Quirk, Robert U Muller, and John L Kubie. The firing of hippocampal place cells in the dark depends on the rat's recent experience. *The Journal of Neuroscience*, 10(6):2008–2017, 1990.

[31] Francesca Sargolini, Marianne Fyhn, Torkel Hafting, Bruce L McNaughton, Menno P Witter, May-Britt B Moser, and Edvard I Moser. Conjunctive representation of position, direction, and velocity in entorhinal cortex. *Science*, 312(5774):758–762, 2006.

[32] Trygve Solstad, Charlotte N Boccara, Emilio Kropff, May-Britt B Moser, and Edvard I Moser. Representation of geometric borders in the entorhinal cortex. *Science*, 322(5909):1865–1868, 2008.

[33] Hanne Stensola, Tor Stensola, Trygve Solstad, Kristian Froland, May-Britt B Moser, and Edvard I Moser. The entorhinal grid map is discretized. *Nature*, 492(7427):72–78, 2012.

[34] Angus Stevenson. *Oxford Dictionary of English.* Oxford University Press, 3 edition, 2013.

[35] Peter S Swain and Andre Longtin. Noise in genetic and neural networks. *Chaos*, 16(2):26–101, 2006.

[36] Edward C Tolman. Cognitive maps in rats and men. *Psychological Review*, 55, 1948.

[37] Case H Vanderwolf. Hippocampal electrical activity and voluntary movement in the rat. *Electroencephalography and Clinical Neurophysiology*, 26(4):407–418, 1969.

[38] R Wehner. Desert ant navigation: how miniature brains solve complex tasks. *Journal of Comparative Physiology A*, 189(8):579–588, 2003.

[39] Adam C Welday, I Gary Shlifer, Matthew L Bloom, Kechen Zhang, and Hugh T Blair. Cosine directional tuning of theta cell burst frequencies: evidence for spatial coding by oscillatory interference. *The Journal of Neuroscience*, 31(45):16157–16176, 2011.

[40] J M Williams and B Givens. Stimulation-induced reset of hippocampal theta in the freely performing rat. *Hippocampus*, 13(1):109–116, 2003.

[41] Eric A Zilli and Michael E Hasselmo. Coupled noisy spiking neurons as velocity-controlled oscillators in a model of grid cell spatial firing. *The Journal of Neuroscience*, 30(41):13850–13860, 2010.