LETTER Communicated by Sebastian Seung

Computing with Continuous Attractors: Stability and Online Aspects

Si Wu
siwu@sussex.ac.uk
Department of Informatics, University of Sussex, Brighton, U.K.

Shun-ichi Amari
amari@brain.riken.go.jp
Laboratory for Mathematical Neuroscience, RIKEN Brain Science Institute, Tokyo, Japan

Two issues concerning the application of continuous attractors in neural systems are investigated: the computational robustness of continuous attractors with respect to input noises and the implementation of Bayesian online decoding. In a perfect mathematical model for continuous attractors, decoding results for stimuli are highly sensitive to input noises, and this sensitivity is the inevitable consequence of the system's neutral stability. To overcome this shortcoming, we modify the conventional network model by including extra dynamical interactions between neurons. These interactions vary according to the biologically plausible Hebbian learning rule and have the computational role of memorizing and propagating stimulus information accumulated with time. As a result, the new network model responds to the history of external inputs over a period of time, and hence becomes insensitive to short-term fluctuations. Also, since dynamical interactions provide a mechanism to convey the prior knowledge of stimulus, that is, the information of the stimulus presented previously, the network effectively implements online Bayesian inference. This study also reveals some interesting behavior in neural population coding, such as the trade-off between decoding stability and the speed of tracking time-varying stimuli, and the relationship between neural tuning width and the tracking speed.

1 Introduction

Recent studies on neural population coding have revealed that continuous stimuli, such as orientation, moving direction, and the spatial location of objects, are likely to be encoded as continuous attractor in neural systems (Amari, 1977; Georgopoulos, Kalaska, Caminiti, & Massey, 1982; Maunsell & Van Essen, 1983; Wilson & McNaughton, 1993; Rolls, Robertson, & Georges-François, 1995; Ben-Yishai, Lev Bar-Or, & Sompolinsky, 1995; Zhang, 1996;

Seung, 1996; Hansel & Sompolinsky, 1998; Taube, 1998; Deneve, Latham, & Pouget, 1999; Wang, 2001; Wu, Amari, & Nakahara, 2002; Stringer, Trappenberg, Rolls, & Aranjo, 2002; Brody, Romo, & Kepecs, 2003; Trappenberg, 2003). The notion of continuous attractor emphasizes that the steady states of the system, which encodes stimulus values, form a continuous parameter space on which the system is neutrally stable. This representation-memory structure contrasts with that of discrete attractors, such as the Hopfield model (Hopfield, 1984), where stimulus values are supposed to be stored as discrete points in the state space. A simple illustration of the structural difference between continuous and a discrete attractors is shown in Figure 1.

The key property of continuous attractors that distinguishes themselves from the discrete ones is the neutral stability of the system. Intuitively, neutral stability implies that there is no resistance along the attractor space, and as a result, the system can change state rather easily under external drives. This property is crucial for neural systems to carry out many important computation tasks, such as tracking a moving object or navigating in space, in which decoding a time-varying stimulus in real time is essential. For systems having discrete attractors, it will be extremely difficult to accomplish these tasks, as the systems have to overcome the associated energy barrier for each state updating. Another good property of continuous attractors, which may not be that obvious, is that they provide a framework for reading out stimuli by using a simple and efficient strategy called *template matching* (details are introduced in section 2.2.2) (Pouget, Zhang, Deneve, & Latham, 1998; Deneve et al., 1999; Wu et al., 2002).

The advantages of continuous attractors, and their associated roles on brain functions, have been widely studied in the literature. An issue, however, that constantly puzzles computational neuroscientists is the instability of continuous attractors. Two aspects of instability are concerned. One is the potential susceptibility of the attractor structure with respect to the imprecision of network components. This takes into account the fact that neuronal synapses in reality may not be as perfect as those required mathematically for maintaining a continuous attractor (Tsodyks & Sejnowski, 1995; Zhang, 1996; Seung, Lee, Reis, & Tank, 2000; Wang, 2001). The other concerns the robustness of network computation with respect to input noises. This takes into account the fact that continuous attractors are intrinsically unstable along the attractor space and that a little fluctuation in input may dramatically modify the decoding result (Amari, 1977; Wang, 2001; Brody et al., 2003). Consider the ubiquity of noise in biological systems a computational model of such instability is essentially useless. Thus, understanding the stability of continuous attractors is of critical importance for us to understand its applications in brain functions. In this study, we mainly focus on how computational robustness of continuous attractors is achieved, and do not consider their structural stability.

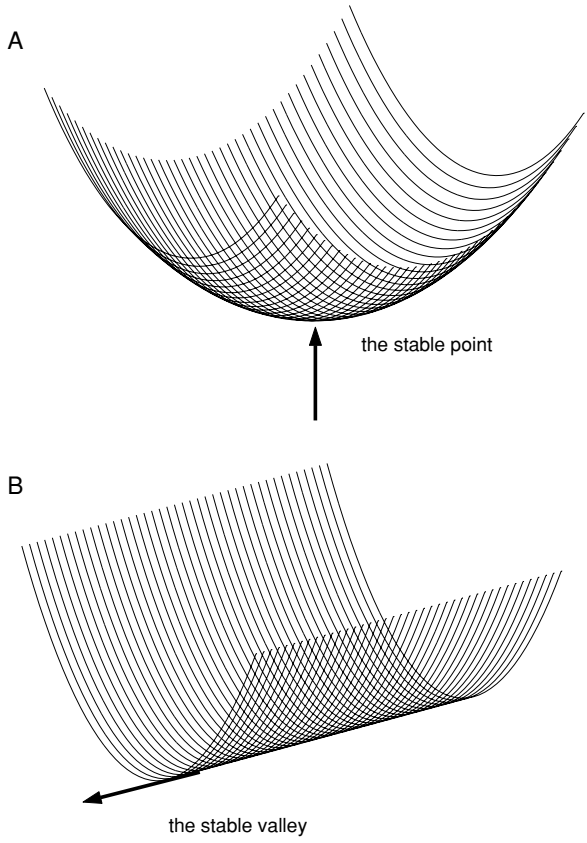

Figure 1: An illustration of structural difference between discrete and continuous attractors. (A) An example of a discrete point attractor. The system is stable only at the bottom of the bowl. (B) An example of line attractor, the one-dimensional version of continuous attractor. The system is stable at any point on the one-dimensional valley and can move easily along the valley under external drive.

Apart from exploring computational robustness, this study also investigates the implementation of Bayesian online decoding in continuous attractors. A significant feature of biological computation is that its input is continuous in time and highly fluctuating (van Vreeswijk & Somplinsky, 1996; Natschlager, Maass, & Markram, 2002). In such a scenario, online computation, in terms of extracting stimulus values in real time and adjusting or modifying them constantly, is the natural solution for efficient information processing. It is far from clear how online decoding is realized in neural systems. Technically, to achieve online computation, the core is to

have a mechanism that can store and propagate input information accumulated in time, so that this information can be used as prior knowledge to enhance further computation. Mathematically, this is expressed as Bayesian inference. In this work, we use continuous attractors as a prototype model for neural computation to explore the possible mechanisms for achieving Bayesian inference in neural systems.

In summary, this study has two main goals: to explore the robustness and online aspects of computation with continuous attractors. For simplicity, we consider only a simple abstract network model for continuous attractors, as is conventionally done in the literature (Amari, 1977; Ben-Yishai et al., 1995; Zhang, 1996; Deneve et al., 1999; Wu et al., 2002; Wu, Chen, Niranjan, & Amari, 2003). This model captures the essential features of how continuous attractors are maintained in neural systems and is consistent with our goal of exploring their general properties. Specifically, we concentrate on analyzing linear attractors, the one-dimensional version of continuous attractors. The results are applicable in general cases.

It turns out that the two goals can be achieved under the same neural mechanism. In the conventional recurrent network models for line attractors, neuronal interactions are usually considered to be static (i.e., unchanged with time). Here, we assume that apart from the original static components, there are also dynamical ones between neuronal interactions. These dynamical interactions vary according to the biologically plausible Hebbian learning rule and have the computational role of storing and propagating the temporal information of external inputs. The underlying picture is straightforward. Whenever the external input changes, due to either noise or real movement of the stimulus, it generates varied neural activities, which subsequently modify the size of dynamical interactions due to Hebbian learning. Thus, the history of external inputs is imprinted in the time course of dynamical interactions. In terms of implementing online Bayesian decoding, dynamical interactions have the role of conveying the prior knowledge of stimuli, that is, the history of external inputs. In terms of improving the computational robustness of continuous attractors, dynamical interactions enable the system to respond to the history of external inputs over a period of time, and hence suppress short-term fluctuations. Part of the idea that dynamical interactions contribute to implementing Bayesian inference in neural circuitry has been studied by Wu et al. (2003).

While the decoding stability of line attractors is improved by dynamical interactions, a compromise is the delayed response to abrupt stimulus changes. This is also understandable. Essentially, stabilizing the computation of line attractors requires the system to respond to the time-averaged behavior of external inputs. Consider the situation that the stimulus experiences an abrupt change, which unfortunately is indistinguishable from noise in a short time window. Since the system responds to inputs over a period of time, its reaction to the sudden change inevitably will be delayed.

We analyze this trade-off between decoding robustness and tracking speed in detail in this article.

As a by-product, this study also reveals some interesting behavior in neural population coding. For instance, we observe that the tracking speed of line attractors strongly depends on the size of the neural tuning width: the larger the tuning width, the quicker the tracking. This gives us a fresh justification of why the neural tuning width should be large, as seen in the data, an issue of wide concern in the study of population coding (Pouget, Deneve, Ducom, & Latham, 1999; Zhang & Sejnowski, 1999).

The organization of this letter is as follows. In section 2, we introduce a simple recurrent network model for line attractors and review its computational advantages. In section 3, a modified network model that includes dynamical interactions is introduced, and its performance is interpreted from the view of Bayesian inference. In section 4, the performances of the new network model are investigated in detail, which particularly include how computation robustness of line attractors is improved, what the trade-off between decoding stability and tracking speed is, and how tracking speed is affected by the size of neural tuning width. Section 5 presents the overall conclusions and discussions of this work.

2 Computing with Line Attractors

In this section, we introduce a recurrent network model for line attractors and review its two main computational advantages.

2.1 The Model. Let us consider a continuous stimulus x encoded by N neurons. For simplicity, we assume $N \longrightarrow \infty$ and consider a continuous neural field model (returning to finite N, however, is straightforward) (Amari, 1977). We use U_c to denote the internal state of neurons whose preferred stimulus is c and O_c the corresponding neural activity (i.e., the firing rate). The nonlinear relationship between O_c and U_c is defined by

$$O_c = \frac{U_c^2}{1 + \mu \int U_c^2 dc}, \tag{2.1}$$

where the parameter μ is a constant. The denominator on the right-hand side reflects the effect of global inhibition. In effect, this prevents neural activities from diverging to infinity.

The neurons in the network are fully connected. The form of the recurrent interaction is the key that generates the structure of the linear attractor, which is set to

$$W_{c,c'} = \exp^{[-(c-c')^2/2a^2]}, \tag{2.2}$$

where the parameter a is the width of the gaussian function. The form of $W_{c,c'}$ exhibits two features. First, $W_{c,c'}$ is translational invariant, being a function of the difference between neuronal preferred stimuli, $(c - c')$. This symmetry leads to the neutral stability of the system. Second, $W_{c,c'}$ has the gaussian form, which ensures that the steady states of the network are bell-shaped, agreeing with experimental observations.

The dynamics of the network is given by

$$\tau \frac{dU_c}{dt} = -U_c + \int_{-\infty}^{+\infty} W_{c,c'} O_{c'} dc' + I_c, \tag{2.3}$$

where the parameter τ is the time constant of U_c, and I_c is the external input.

It is straightforward to check that with the above recurrent interactions and proper values of μ, when $I_c = 0$, the network accommodates a family of nontrivial stable states, which can be written as[1]

$$\tilde{O}_c = A \exp^{[-(c-z)^2/2a^2]}, \quad \forall z, \tag{2.4}$$

$$\tilde{U}_c = B \exp^{[-(c-z)^2/4a^2]}, \quad \forall z, \tag{2.5}$$

where the coefficients A and B are constants depending on μ, and the variable z is a one-dimensional free parameter. We see that the steady states of network indeed have a bell shape. Since equations 2.4 and 2.5 hold for any value of z, the system is said to have a line attractor.

The value of z indicates the position of the steady state on the attractor space, which is also the network representation of stimulus. In case there is no external input, that is, $I_c = 0$, the system may stay at any point on the attractor, depending on the initial condition of the dynamics. When an external input is presented, the network will be driven to a particular position, with the corresponding value of z reporting the decoding result.

It is worth noting that the parameter a has two functional meanings: it determines the range of neuronal interactions (see equation 2.2), and it defines the width of neural tuning function (see equation 2.4).

2.2 Computational Advantages of Linear Attractor. Based on this model, we now illustrate two main advantages of line attractors: to track stimuli smoothly and to implement template matching.

To mimic real situations more precisely, such as when the stimulus is the orientation or moving direction of objects, we restrict the stimulus in the simulation to be a periodical variable. More exactly, we consider stimulus x (and so does the preferred stimulus of neurons) in the range

[1] An easy way to check that equations 2.4 and 2.5 are the stable states of the network is to substitute them in equations 2.1 and 2.3 and see whether $dU_c/dt = 0$ (and, hence, $dO_c/dt = 0$) holds when $I_c = 0$.

$(-\pi, \pi]$, with $x = \theta$ and $x = 2\pi + \theta$ being the same. Under this condition, the steady states of the network will no longer have the exact gaussian form as in equation 2.4. However, provided that the value of a is not too large, say, $a < \pi$ as considered here, the bell shape of steady states still holds, as confirmed by the simulation.

With the periodic condition, the form of recurrent interactions is adjusted to be

$$W_{c,c'} = \begin{cases} \exp^{[-(c-c')^2/2a^2]} & \text{if } |c - c'| \le \pi \\ \exp^{[-(2\pi - |c-c'|)^2/2a^2]} & \text{if } |c - c'| > \pi \end{cases}. \tag{2.6}$$

2.2.1 Tracking Time-Varying Stimuli Smoothly. One critical advantage of line attractors is their capability to track time-varying stimuli smoothly. For illustration, we consider a challenging situation (Georgopoulos, Taira, & Lukashin, 1993; Ben-Yishai et al., 1995), in which the stimulus experiences an abrupt change, say, jumping from $x = 0$ jumps to $x = h$, for h a small constant. This process can be modeled by setting external input $I_c = \gamma \exp^{-c^2/2a^2}$ initially, with γ a small constant, and then changing I_c suddenly to $I_c = \gamma \exp^{-(c-h)^2/2a^2}$.

Figure 2A records the activities of the network from the onset of the change to the stationary state. It shows that the network can indeed track the abrupt change of stimulus, in the sense that the network representation for stimuli is updated accordingly. In contrast, we also illustrate the typical behavior expected for discrete attractors in the same situation, as shown in Figure 2B. We see that for discrete attractors, the system state is trapped at the original position and is unable to follow the stimulus change.

Figure 2A also exhibits another important characteristic of line attractors: during tracking, the bell shape of population activity is roughly maintained. This property can be illustrated in another way. Let us consider the case of the stimulus being an angle variable (e.g., the orientation or moving direction) and record network representations during the intermediate stages of tracking. The result is shown in Figure 3, which demonstrates that the network state rotates in a seamless way from the initial position to where the stimulus value has changed. This phenomenon is actually a unique feature of continuous attractors, not shared by discrete ones (Ben-Yishai et al., 1995). It tells us that the state change of continuous attractors must follow paths defined by the attracting space. For instance, if the stimulus changes from 0 to h, the state of line attractors must go through all the intermediate values between 0 and h. For discrete attractors, there is no such restriction, and the system in principle can update its state discontinuously from 0 to h, though this process may need extra effort. This property of seamless rotation has been identified as important evidence for the existence of line attractors in neural systems (Georgopoulos et al., 1993).

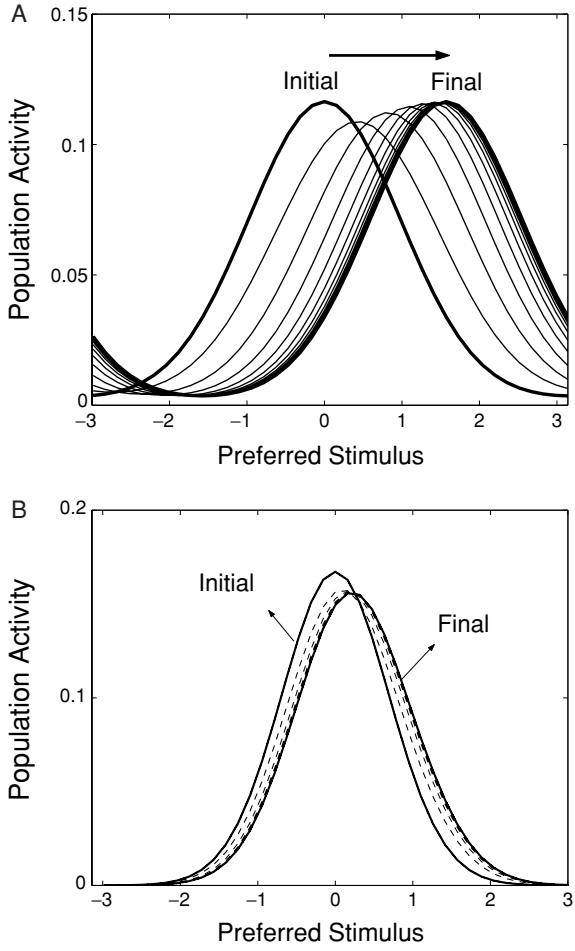

Figure 2: An illustration of network response in the situation when the stimulus experiences an abrupt change. The population activity of the network is drawn every 20τ time units. Neurons are labeled by their preferred stimuli. In the simulation, the stimulus value is abruptly changed from 0 to 0.5π. The parameters used are $N = 40, a = 1, \tau = 1, \mu = 0.5$ and $\gamma = 0.05$. (A) The line attractor. (B) The discrete attractor. The discrete attractor is constructed by adding a small amount of position-dependent component, $0.1e^{-(c^2+c'^2)^2}$, to the recurrent interactions, equation 2.2, so that the neutral stability of the system is broken.

2.2.2 Decoding Stimulus by Template Matching. Another advantage of linear attractors is their effectiveness in reading out stimuli from a noise environment. Let us consider an external input that is noisy and modeled by $I_c = \gamma \exp^{[-(c-x)^2/2a^2]} + \epsilon_c$, where x denotes the true stimulus and ϵ_c a

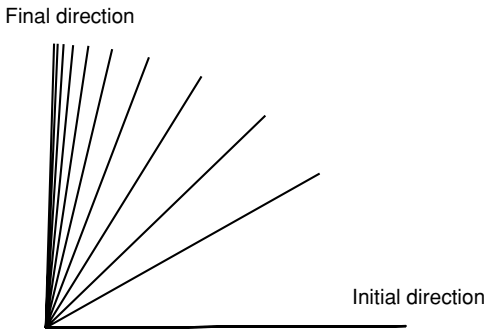

Figure 3: An alternative way to illustrate the tracking process shown in Figure 2A. The network representation is represented by a direction vector.

random variable. The goal of the neural estimator is to infer the value of x from the input I_c. Provided that I_c is a constant during decoding (though this is not plausible in practice), it will drive the network to the position on the attractor space that has the maximum overlap with I_c (Pouget et al., 1998; Deneve et al., 1999; Wu et al., 2002), that is,

$$\hat{z} = \text{argmax}_z \int_{-\pi}^{\pi} I_c \tilde{O}_c(z) dc, \tag{2.7}$$

where $\hat{z}$ denotes the final position of the network state, that is, the estimation of stimulus, and $\tilde{O}_c(z)$ is given by equation 2.4. This decoding operation is called *template matching*, in the sense that the tuning function, $\{\tilde{O}_c\}$, is used as the template to match I_c. This process is illustrated in Figure 4.

To implement template matching, the neutral stability of the line attractor plays the key role, which guarantees that the template can be moved freely in order to fit I_c. For the decoding performance of template matching, it has been proved that template matching is equivalent to maximum likelihood inference (MLI) based on an unfaithful model that neglects the correlation between neuronal activities (Wu, Nakahara, & Amari, 2001). In general cases, template matching outperforms center of mass (COM) (or population vector equivalently). Compared with MLI that uses all information of the encoding process, template matching has the advantage of being much simpler.

3 Line Attractors with Dynamical Interactions

A possible difficulty for applying line attractors in practice is its sensitivity to input noise, the inevitable consequence of the attractor's neutral stability.

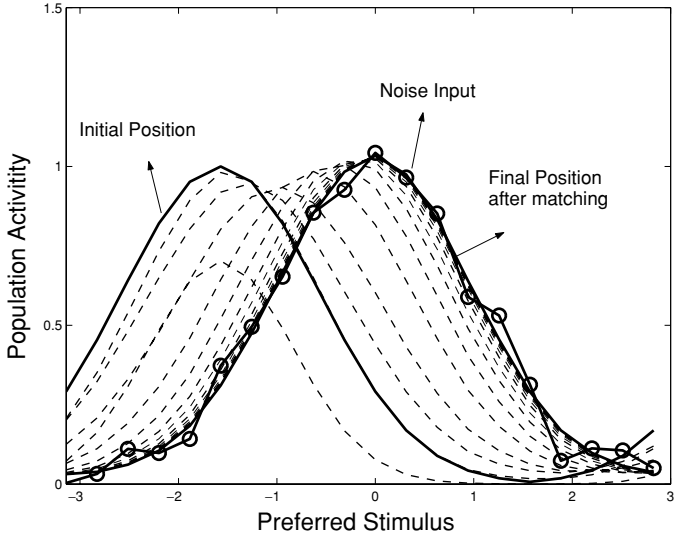

Figure 4: An illustration of the template-matching operation. The initial position of the template is at $z = -0.5\pi$. The true stimulus value is at $x = 0$. The diagram shows how the template is driven by the external input to a new position that has the maximum overlap with the noisy input.

Input noises generate a random drift, constantly fluctuating along the attractor space,[2] which makes decoding unreliable. Thus, to improve the robustness of decoding, the neural system must have a mechanism to suppress this sensitivity.

Consider that neural systems have no extra source of information on stimulus except from the one of external input; an efficient way to increase computational robustness is to let the system respond to the time-averaged behavior of external input, so that short-term fluctuations can be suppressed. To achieve this, the essence is to have a mechanism that can store and propagate input information accumulated with time. For the simple abstract model we consider, the form of recurrent interactions determines the type of operation performed by the network. We therefore expect that by properly adjusting recurrent interactions, the neural mechanism we want can be achieved.

[2] Only in the extreme case when the input noise is constant over time, the driving force generated along the valley is vanishing after the system is relaxed. This is the case considered in Pouget et al. (1998), Deneve et al. (1999), and Wu et al. (2002).

3.1 The New Network Model. The new network model we consider has the following form

$$\tau \frac{dU_c}{dt} = -U_c + \int_{-\pi}^{\pi} (W_{c,c'} + w_{c,c'}) O_{c'} dc' + I_c, \tag{3.1}$$

$$\tau_w \frac{dw_{c,c'}}{dt} = -w_{c,c'} + \eta O_c O_{c'}, \tag{3.2}$$

where we assume that the recurrent interaction consists of two parts: the original static component, $W_{c,c'}$, given by equation 2.1, and the new dynamical one, $w_{c,c'}$, which varies according to the Hebbian learning rule. The parameters τ_w and η are the time constant and the learning rate of $w_{c,c'}$, respectively. From now on, we consider the fluctuations of external input I_c changing over time. Without loss of generality, we consider the fluctuations that are updated every T time units, that is, the frequency of noise is $1/T$.

The dynamical behavior of the above network is governed by three timescales: (1) the frequency of input fluctuations, given by $1/T$; (2) the converging speed of neural activity when I_c and $w_{c,c'}$ are fixed, which is determined by $1/\tau$; and (3) the response speed of dynamical interactions with respect to the change of neural activities, which is controlled by τ_w and η.

We are particularly interested in the parameter region when $T \gg \tau$ and $\tau_w \geq T$. The first requirement ensures that the updates of network states can catch up to the input change. Computationally, this is the only meaningful case, since any input information varying more quickly than $1/\tau$ is undetectable by the network. The second requirement ensures that dynamical interactions will memorize the history of external input over a sufficiently long time.

3.2 Statistic Interpretation of the Network Performance. In general, it is difficult to analytically solve the dynamics of above network model, whose solution largely depends on the details of the external input. In order to get an intuitive understanding of the role of dynamical interactions, we first consider a special case in which the input fluctuations are sufficiently small. This condition allows us to approximate the continuous equation 3.2 as an equation discrete in time. More specifically, we approximate $w_{c,c'}$ to be a constant during each period of T when I_c is fixed. This takes into account the fact that within each period, the contribution due to the variation of $w_{c,c'}$ on the change of network states is a small quantity of a higher order when compared with that due to the change of input. As compensation of this approximation, $w_{c,c'}$ is updated at the end of each period according to

$$w_{c,c'}(m) \approx \beta w_{c,c'}(m-1) + (1-\beta)\eta \tilde{O}_c(m-1)\tilde{O}_{c'}(m-1), \tag{3.3}$$

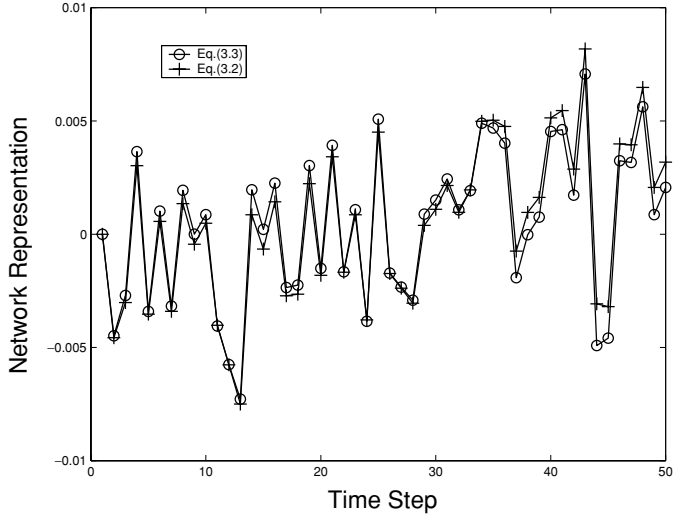

Figure 5: Confirming the plausibility of the approximation 3.3. The network performances based on equations 3.2 and 3.3 are compared. The true stimulus $x = 0$. The unit of each time step is T. The network estimation of stimulus at each time step is shown. The parameters $T = 20\tau$ and $\tau_w = 100\tau$, which gives $\beta = e^{-T/\tau_w} \approx 0.82$. Gaussian random noise is added in the input, which has zero mean and variance 2×10^{-3}. The other parameters are $N = 40$, $a = 1$, $\tau = 1$, $\mu = 0.5$, $\gamma = 0.05$, and $\eta = 10$.

where m labels the time step of duration T, $w_{c,c'}(m)$ the approximated value of the dynamical interaction at the step m, and $\tilde{O}_c(m-1)$ the stable state of neurons at step $m-1$. The coefficient β is given by $\beta = \exp^{[-T/\tau_w]}$. Equation 3.3 can be seen as the solution to equation 3.2 under the approximation $O_c(t) = \tilde{O}_c(m-1)$.

The value of $w_{c,c'}(m)$ consists of two parts. The first one is inherited from the previous time step, and the second is the newly learned result. The plausibility of this approximation is confirmed by the simulation in Figure 5.

By iteration, we further write down the dependence of $w_{c,c'}(m)$ on all previous neural activities,

$$w_{c,c'}(m) \approx (1-\beta)\eta[\tilde{O}_c(m-1)\tilde{O}_{c'}(m-1) + \beta\tilde{O}_c(m-2)\tilde{O}_{c'}(m-2) + \beta^2\tilde{O}_c(m-3)\tilde{O}_{c'}(m-3) + \cdots], \tag{3.4}$$

which displays how the history of input information is imprinted on the time course of dynamical interactions.

With the above approximation, the network estimation at step m is calculated to be (see the appendix),

$$\hat{z}(m) = \text{argmax}_z \left[\int_{-\pi}^{\pi} \tilde{O}_c(z) I_c(m) dc + (1-\beta)\eta \sum_{k=1} \beta^{k-1} \left(\int_{-\pi}^{\pi} \tilde{O}_c(z) \tilde{O}_c(m-k) dc \right)^2 \right]. \quad (3.5)$$

From the above equation, we see that the network estimation is determined by two parts. The first corresponds to the conventional template matching with respect to the instant external input (compare with equation 2.7), and the second with the summation of contributions from all previous neural activities, weighted according to their temporal proximity. Thus, equation 3.5 can be seen as implementing an online Bayesian inference, where the first term corresponds to MLI and the second to the contribution of prior knowledge.[3]

Now, the role of dynamical interactions becomes clear. It serves to store and propagate the input information being accumulated with time.

4 The Performances of the New Network Model

Now we investigate the detailed performance of the new network model.

4.1 Reading-Out Stimulus Smoothly and Accurately. To confirm the analysis in section 3.2, we carry out the following simulation experiment. We consider a situation in which the true stimulus x is fixed, whereas the external input fluctuates over time, which is written as

$$I_c(m) = \gamma e^{[-(c-x)^2/2a^2]} + \epsilon_c(m), \quad (4.1)$$

where $I_c(m)$ is the input value at the time step m and $\epsilon_c(m)$ the corresponding noise.

[3] To formally illustrate that equation 3.5 corresponds to Bayesian inference (more specifically, the implementation of maximum a posteriori), we may need to formulate the second term as the logarithm of a prior distribution of stimulus. It turns out, however, that in general situations, this distribution is complex and does not have a simple form. Only in the extreme case, as considered in Wu et al. (2003), which corresponds to β being sufficiently small here, this distribution can be formulated as a gaussian prior centered at the network estimation of the previous step (but in this case, the learning rate needs to be subtly adjusted in order to efficiently use the prior knowledge). For details, refer to Wu et al. (2003).

Without loss of generality, we consider $\epsilon_c(m)$ being independent gaussian white noise satisfying

$$\langle \epsilon_c(m) \rangle = 0, \tag{4.2}$$

$$\langle \epsilon_c(m)\epsilon_{c'}(m) \rangle = \sigma^2 \delta(c - c'), \tag{4.3}$$

$$\langle \epsilon_c(m)\epsilon_c(m') \rangle = \sigma^2 \delta(m - m'), \tag{4.4}$$

where σ denotes the noise strength and the symbol $\langle \cdot \rangle$ the average over many trials. The condition (4.3) implies that noises are independent between the ensemble units and the condition (4.4) noises are uncorrelated over time.

To check the susceptibility of network decoding with respect to input fluctuations, we measure its decoding error, defined by $\langle (\hat{x} - x)^2 \rangle$, and compare the obtained result with that of the original network model without dynamical interactions. The typical performances of the two networks are shown in Figure 6. We see that the new network becomes much more robust with respect to input noises. The extent of improvement depends on the choice of β (or τ_w) and η, that is, how much history of the external input is memorized by the dynamical interactions. For the case in Figure 6, the decoding error of the new network is down to 6×10^{-3}, compared with 2×10^{-2} of the original model.

4.2 Retaining the Capacity of Smooth Tracking. A major reason for neural systems to adopt continuous attractors is that they provide the capacity of tracking time-varying stimuli smoothly. Therefore, it is important to check whether this property is retained in the new network model.

Again, as in the case of Figure 2A, we consider a situation where the stimulus experiences an abrupt change. The typical behavior of the new network is illustrated in Figure 7, which confirms that the property of smooth tracking indeed holds.

Intuitively, one may suspect that since $w_{c,c'}$ has now broken the translational invariance of the recurrent interactions (consider a situation when a fixed input has been presented for a sufficiently long time; the value of $w_{c,c'}$ becomes concentrated around that input), why it does not destroy the property of smooth tracking (see the example in Figure 2B). The explanation is that $w_{c,c'}$ is dynamical here, which also evolves under the drive of external input.

4.3 The Trade-Off Between Decoding Stability and the Speed of Tracking. We do, however, observe a difference in performance between the new and old network models: the reaction speed to abrupt stimulus change is slower in the new model. The reaction time, whose inverse defines the response speed, is measured from the moment the change occurs to when the

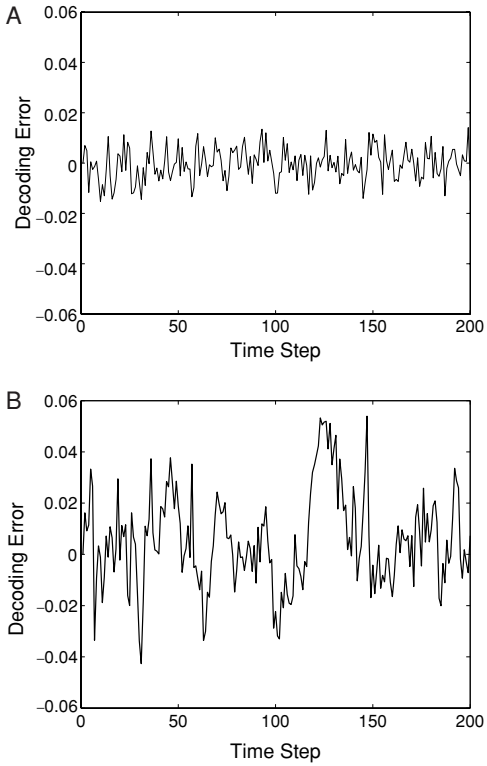

Figure 6: A comparison of the decoding performances of the two network models with and without dynamical interactions. The noise strength $\sigma^2 = 0.01$. (A) New network model. The parameters are $\beta = 0.8$ and $\eta = 10$. (B) Original network model.

system catches up to the change. This phenomenon is not a surprise to us, which is due to the fact that the neural system responds to the time-averaged behavior of the external input. Thus, there is a trade-off between decoding robustness of line attractors and their speed of tracking. It is conceivable that the longer the network memorizes the input history, the more robust the network decoding will be and the more likely the tracking speed will be delayed.

The parameters that determine how much history of the external input is to be memorized by dynamical interactions are β and η. From eqution 3.5, we see that the larger the value of η, the more the network estimation depends on previous neural activities. Thus, the tracking speed should decrease with η (or equivalently, the reaction time increases with η), as confirmed

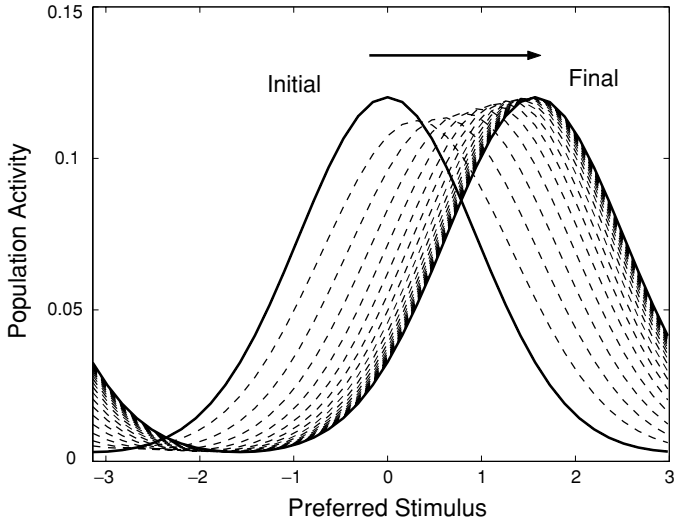

Figure 7: Illustration of the tracking capacity of the new network model. The input condition is as the same as in Figure 2. The parameters for dynamical interactions are $\beta = 0.8$ and $\eta = 10$.

in Figure 8A.[4] To check the trade-off on decoding robustness, we measure decoding accuracy versus η. The decoding accuracy is calculated when the true stimulus is fixed (only in this case can the accuracy be easily calculated by the mean square error). We observe that the decoding accuracy increases with η, as expected. For $\beta = 0.8$ and the noise strength $\sigma^2 = 0.01$, decoding errors are calculated to be 9×10^{-3}, 6×10^{-3}, and 4×10^{-3} for $\eta = 5, 10$, and 15, respectively.

Similarly, the trade-off between robustness and tracking speed can also be displayed based on their dependence with β. The larger the value of β, the more information about previous neural activities is memorized by dynamical interactions, and, hence, the longer delay in tracking can be expected. This is confirmed in Figure 8B. As a trade-off, decoding robustness increases with β. For $\eta = 10$ and $\sigma^2 = 0.01$, decoding errors are calculated to be 8^{-3}, 6×10^{-3}, and 2.5×10^{-3} for $\beta = 0.7, 0.8$, and 0.9, respectively.

[4] One may note that in both Figures 8 and 9, there is a sudden jump in the network representation immediately after the stimulus change. Mathematically, this is due to the sudden change of input value, as is evident by the fact that when we vary the size of γ in I_c, the magnitude of the jump will also change. It is not clear whether this implies certain biological behavior.

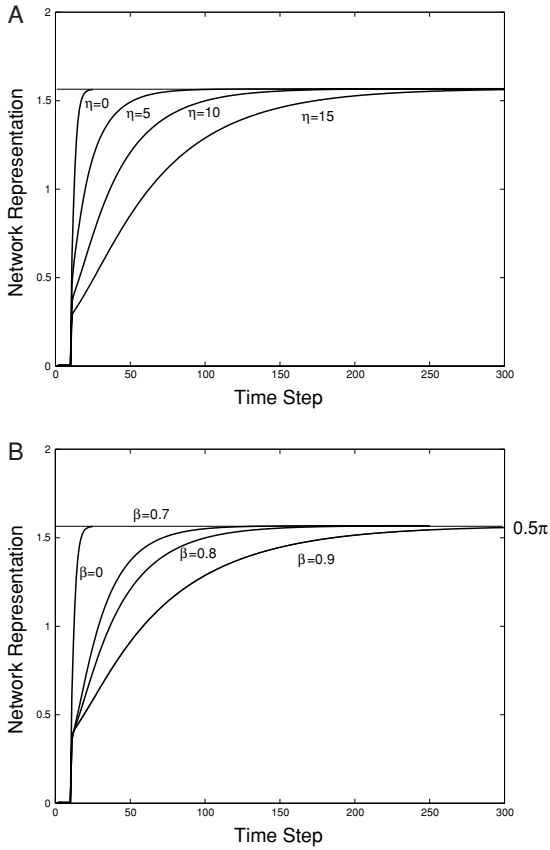

Figure 8: Illustration of the dependence of tracking speed on the size of η and β. In the simulation, the stimulus value is abruptly changed from 0 to 0.5π. The time course of the network representation during the tracking is shown. (A) For different values of η, $\beta = 0.8$. $\eta = 0$ corresponds to the situation of no dynamical interactions. (B) For different values of β, $\eta = 10$. $\beta = 0$ corresponds to the situation of no dynamical interactions.

4.4 Reaction Time vs. the Magnitude of Abrupt Stimulus Change. It is also valuable to see the relationship between the amount of delay and the size of the abrupt stimulus change. Figure 9A shows that the amount of delay increases with the jump size. This finding qualitatively agrees with the experimental observation (Georgopoulos & Massey, 1987; Georgopoulos et al., 1993).

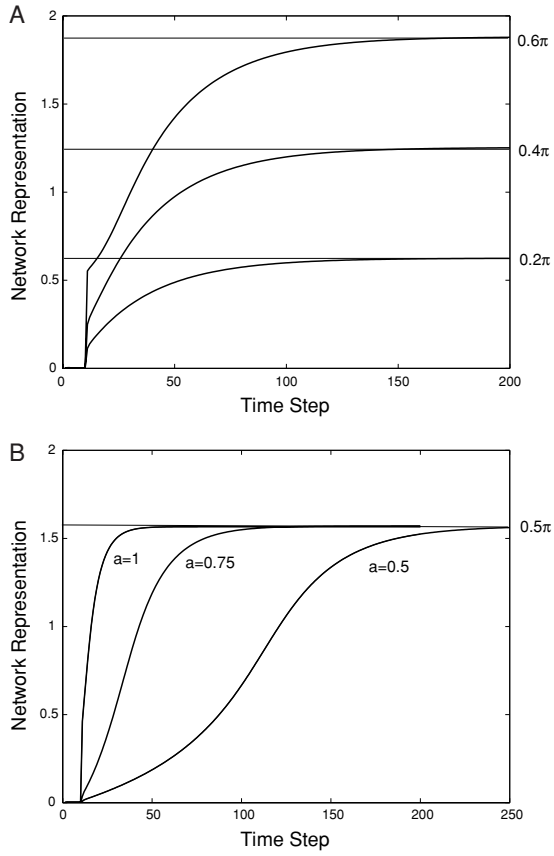

Figure 9: (A) Illustration of the dependence of tracking time on the size of abrupt stimulus change. In the simulation, the stimulus value abruptly jumps from 0 to 0.2π, 0.4π, and 0.6π, respectively. (B) Illustration of the dependence of tracking on the width of tuning function. In the simulation, the stimulus value abruptly changes from 0 to 0.5π.

4.5 Tracking Speed vs. Neural Tuning Width. Another interesting behavior we observe is the dependence of the tracking speed on the width of tuning function. Figure 9B shows that the tracking speed increases with the tuning width a. This is understandable. Intuitively, tracking is to move the bump of population activity from the initial position to where the stimulus has changed (see Figure 7), and this operation is conducted through neuronal recurrent interactions. Therefore, it can be expected that the broader the range of neural interactions, the quicker the bump can move. From equation 2.1, we see that the tuning width a controls the range of

neuronal interactions. Thus, larger tuning width implies quicker tracking speed. Notably, this property also holds for the network without dynamical interactions.

This finding gives us a new justification for why the neural tuning width is large, a long-standing debate in the field of population coding (Pouget et al., 1999; Zhang & Sejnowski, 1999). Through calculating the Fisher information (whose inverse, the Cramér-Rao bound, defines the optimal accuracy for any unbiased estimator to achieve), Zhang and Sejnowski (1999). found that smaller tuning width actually leads to higher decoding accuracy. This seems to be in contrast to the experimental finding that the neural tuning width is often very large and suggests that apart from decoding accuracy, other factors also influence the construction of neural population code. One such factor is believed to be the speed of computation (Brunel & Nadal, 1998; Bethge, Rotermund, & Pawelzik, 2002). The idea is as follows. Large tuning width implies that more neurons can be active shortly after the onset of stimulus, and hence more neurons are involved in decoding in a short time window. This helps to average out fluctuations in the activities of individual neurons, and hence increases the decoding accuracy in a short time window.[5] The finding of Figure 9B gives us another justification on large tuning width: it increases the tracking speed of neural systems, a critical property needed for carrying out many computational tasks.

5 Conclusions and Discussions

This study has investigated two important issues that arise when applying continuous attractors in neural systems. One issue concerns the computational robustness of continuous attractors with respect to input noises, a consequence of the system's neutral stability. The other concerns the implementation of Bayesian online coding, an advanced strategy to read out stimulus accurately from a noisy environment. It turns out that both aspects can be resolved under the same neural mechanism, that is, to include dynamical interactions of proper form between neurons. These dynamical interactions evolve according to the biologically plausible Hebbian learning rule and have the computational role of storing and propagating input information accumulated with time. In terms of stabilizing the computation of continuous attractors, dynamical interactions enable the system to respond to the history of external input over a period of time and, hence, suppress short-term fluctuations. In terms of realizing Bayesian online inference, dynamical interactions have the effect of conveying the prior knowledge of

[5] In a short time window, MLI is not asymptotically efficient; instead, its decoding error satisfies the Cauchy distribution. The Cramér-Rao bound is not achievable. This is the reason that the analysis based on the Fisher information fails in this case (Wu & Dayan, unpublished result).

stimulus, which, in the coding paradigm of our concern, is the temporal information of external inputs.

Understanding the stability of continuous attractors, on either its computational or structure robustness, has been an active topic in the research of neural information processing (e.g., see (Wang, 2001; Brody et al., 2003)). Apart from the one proposed here, another potential mechanism to improve the robustness of attractor computation is to consider the multiple stability of neural responses (Camperi & Wang, 1998; Koulakov, Raghavachari, Kepecs, & Lisman, 2002). Because of multiple stability, the system will not respond to small variations of input signals, and hence achieves a certain level of robustness. This mechanism is different from the one proposed in this work. In our consideration, it is the response of the system to the time-averaged behavior of input that increases the robustness of computation. The beneficial point of our mechanism is that its realization is naturally associated with Bayesian online inference. In other words, neural systems do not simply ignore noise in order to maintain stability; instead, they actively use this information to enhance decoding (note that in a noisy environment, it is the time dependence of input fluctuations that gives us the information about the stationary value of the stimulus). Nevertheless, these two mechanisms are not necessarily contradictory. They may be applied in different conditions, for example, for noises of different sizes or different timescales. Further studies are needed to clarify this point.

Exploring the neural mechanism for implementing Bayesian inference is the other main goal of this work. It is widely believed that Bayesian inference should be used in neural systems to cope with typical inputs in the natural environment that are both noisy and highly structured (see, e.g., Mumford, 1992; Dayan, Hinton, Neal, & Zemel, 1995; Zemel, Dayan, & Pouget, 1998; Zhang, Ginzburg, McNaughton, & Sejnowski, 1998; Baddley, Hancock, & Foldiak, 2000; Rao, Olshausen, & Lewicki, 2002; Wu et al., 2003; Lee & Mumford, 2003; Rao, 2004). To implement Bayesian inference, the core is to have a mechanism that can store and propagate the prior knowledge of stimulus. In this work, we investigate this issue for the case of using line attractors to decode stimulus. For the coding paradigm of concern, the prior knowledge of stimulus is the temporal information of external inputs, which is critically needed for suppressing short-term fluctuations. Interestingly, it turns out that Hebbian learning, the fundamental principle that underlies the modification of neural interactions, serves to convey this prior knowledge. It converts the temporal information of external inputs into spatially distributed patterns of neuronal interactions, which subsequently influence the computation in network. Also, the amount of temporal information to be memorized by the network can be adaptively controlled through adjusting the decay and learning speed of dynamical interactions. This finding may have important implications for our understanding of neural coding. It suggests that Hebbian learning can also play an active role in information retrieval, not merely in learning new knowledge (Bi & Poo, 2001).

Dynamical recurrent interactions can also have other contributions in computation associated with continuous attractors. For instance, Zhang (1996) investigated a line attractor model for head direction representation, in which a key operation is to integrate the self-motion signal of object with the observer-centered one, so that the world-centered representation for head direction can be achieved. Zhang found that this computation can be carried out by including properly varied recurrent interactions. Stringer et al. (2002) further show that these rapidly varying interactions can be generated by Hebbian learning, sharing the same mechanism as proposed here for stabilizing the computation of line attractors and implementing Bayesian online decoding. It will be interesting to see how these studies can be combined to give a full picture of the roles of dynamical interactions in attractor computation.

An obvious direction to extend the current work is to include more detailed structures of biological systems. Recent research has shown that continuous attractors can be readily achieved in spiking neural networks (Trappenberg, 1998; Seung et al., 2000; Gutkin, Laing, Colby, Chow, & Ermentrout, 2001; Degris, Brunel, Sigaud, & Arleo, in press; Osan, Curtu, Rubin, & Ermentrout, 2004). Considering the simplicity and plausibility of computational roles of dynamical interactions proposed here, we expect that they are also feasible for spiking neurons. Taking into account the nature of timescale considered here, we expect that the neural basis of dynamical interactions is associated with the short-term activity-dependent plasticity of synapses (Bi & Poo, 2001). These issues will be explored in our future study.

Appendix: Bayesian Interpretation of the Network Performance

Under the approximation that $w_{c,c'}$ is a constant in each time interval when I_c is fixed, the dynamics of the new network model is simplified as

$$\tau \frac{dU_c}{dt} = -U_c + \int_{-\pi}^{\pi} (W_{c,c'} + w_{c,c'}(m)) O_{c'} dc' + I_c(m), \tag{A.1}$$

where $w_{c,c'}(m)$ is given by equation 3.3.

Since we are interested only in the case when noise fluctuations are sufficiently small, the above equation can be further approximated as

$$\tau \frac{dU_c}{dt} \approx -U_c + \int_{-\pi}^{\pi} W_{c,c'} O_{c'} dc' + I'_c(m), \tag{A.2}$$

with

$$I'_c(m) = I_c(m) + \int_{-\pi}^{\pi} w_{c,c'}(m) \tilde{O}_{c'}(m) dc'. \tag{A.3}$$

To get equation A.2, the condition $\int_{-\pi}^{\pi} w_{c,c'}(m)O_{c'}dc' \approx \int_{-\pi}^{\pi} w_{c,c'}(m)\tilde{O}_{c'}(m)dc'$, is used, where $\tilde{O}_c(m)$ is the stable state of neurons in the step m. This approximation takes into account the fact that $\int_{-\pi}^{\pi} w_{c,c'}(m)(O_{c'} - \tilde{O}_{c'}(m))dc'$ is a small quantity of higher order when compared with the input fluctuations.

Comparing equation A.2 with equation 2.3, we see the only difference is that I_c is replaced by $I'_c(m)$. Thus, in effect, the contribution of dynamical interactions is equivalent to an extra "input" component to neurons.

When $I'_c(m)$ is sufficiently small, the case we consider, the solution of equation A.2, that is, the estimation of the network, is given by Wu et al. (2003),

$$\hat{z}(m) = \text{argmax}_z \int_{-\pi}^{\pi} \tilde{O}_c(z)I'_c(m)dc. \tag{A.4}$$

Further, using equation 3.4, we get

$$\hat{z}(m) = \text{argmax}_z \left[\int_{-\pi}^{\pi} \tilde{O}_c(z)I_c(m)dc + (1-\beta)\eta \sum_{k=1}^{\infty} \beta^{k-1} \left(\int_{-\pi}^{\pi} \tilde{O}_c(z)\tilde{O}_c(m-k)dc \right)^2 \right]. \tag{A.5}$$

The above equation consists of two terms. The first one corresponds to the contribution from the current input, and the second one to the contributions from previous neural activities.

Acknowledgments

We acknowledge valuable discussions with Thomas Trappenberg, Pulin Gong, Sophie Deneve, Kosuke Hamaguchi, and Peter Dayan. We also thank K. Y. Michael Wong for improving the writing of this letter.

References

Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, *27*, 77–87.

Baddley, R., Hancock, P., & Foldiak, P. (2000). *Information theory and the brain*. Cambridge: Cambridge University Press.

Ben-Yishai, R., Lev Bar-Or, R., & Sompolinsky, H. (1995). Theory of orientation tuning in visual cortex. *Proc. Natl. Acad. Sci. USA*, *92*, 3844–3848.

Bethge, M., Rotermund, D., & Pawelzik, K. (2002). Optimal short-term population coding: When Fisher information fails. *Neural Computation*, *14*, 2317–2351.

Bi, G., & Poo, M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annu. Rev. Neurosci.*, *24*, 139–166.

Brody, C. D., Romo, R., & Kepecs, A. (2003). Basic mechanisms for graded persistent activity: Discrete attractors, continuous attractors, and dynamic representations. *Current Opinion in Neurobiology*, *13*, 204–211.

Brunel, N., & Nadal, J.-P. (1998). Mutual information, Fisher information, and population coding. *Neural Computation*, *10*, 1731–1757.

Camperi, M., & Wang, X.-J. (1998). A model of visuospatial short-term memory in prefrontal cortex: Recurrent network and cellular bistability. *J. Comput. Neurosci.*, *5*, 383–405.

Dayan, P., Hinton, G. E., Neal, R. M., & Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, *7*, 889–904.

Degris, T., Brunel, N., Sigaud, O., & Arleo, A. (in press). Rapid response of head direction cells to reorienting visual cues: A computational model. *Neural Computing*.

Deneve, S., Latham, P. E., & Pouget, A. (1999). Reading population codes: A neural implementation of ideal observers. *Nature Neuroscience*, *2*, 740–745.

Georgopoulos, A. P., Kalaska, J. F., Caminiti, R., & Massey, J. T. (1982). On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *J. Neurosci.*, *2*, 1527–1537.

Georgopoulos, A. P., & Massey, J. T. (1987). Cognitive spatial-motor process. *Experimental Brain Research*, *65*, 361–370.

Georgopoulos, A. P., Taira, M., & Lukashin, A. (1993). Cognitive neurophysiology of the motor cortex. *Science*, *260*, 47–52.

Gutkin, B. S., Laing, C. R., Colby, C. L., Chow, C. C., & Ermentrout, B. (2001). Turning on and off with excitation: The role of spike-timing asynchrony and synchrony in sustained neural activity. *J. Computational Neuroscience*, *11*, 121–134.

Hansel, D., & Sompolinsky, H. (1998). Modeling feature selectivity in local cortical circuits. In C. Koch and I. Segev (Eds.), *Methods in neuronal modeling: From synapses to networks*. Cambridge, MA: MIT Press.

Hopfield, J. J. (1984). Neurons with graded responses have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA*, *81*, 3088–3092.

Koulakov, A. A., Raghavachari, S., Kepecs, A., & Lisman, J. E. (2002). Model for robust neural integrator. *Natural Neuroscience*, *5*, 775–782.

Lee, T. S., & Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A*, *20*, 1434–1448.

Maunsell, J. H. R., & Van Essen, D. C. (1983). Functional properties of neurons in middle temporal visual area of the macaque monkey. I. Selectivity for stimulus direction, speed, and orientation. *J. Neurophysiology*, *49*, 1127–1147.

Mumford, D. (1992). On the computational architecture of the neocortex. II. The role of cortico-cortical loops. *Biol. Cybern.*, *66*, 241–251.

Natschlager, T., Maass, W., & Markram, H. (2002). The "liquid computer": A novel strategy for real-time computing on time series [Special issue]. *Foundations of Information Processing of TEIEMATIK*, *8*, 39–43.

Osan, R., Curtu, R., Rubin, J., & Ermentrout, B. (2004). Multiple-spike waves in a one-dimensional integrate-and-fire neural network. *J. Math. Biol.*, *48*, 243–274.

Pouget, A., Deneve, S., Ducom, J., & Latham, P. (1999). Narrow versus wide tuning curves: What's best for a population code? *Neural Computation*, *11*, 85–90.

Pouget, A., Zhang, K., Deneve, S., & Latham, P. E. (1998). Statistically efficient estimation using population coding. *Neural Computation*, *10*, 373–401.

Rao, R. P. (2004). Bayesian computation in recurrent neural circuits. *Neural Computation*, *16*, 1–38.

Rao, R. P., Olshausen, B., & Lewicki, M. (2002). *Probabilistic models of the brain, perception and neural function*. Cambridge, MA: MIT Press.

Rolls, E. T., Robertson, R. G., & Georges-François, P. (1995). The representation of space in the primate hippocampus. *Soc. Neurosci. Abstr.*, *21*, 1494.

Seung, H. S. (1996). How the brain keeps the eyes still. *Proc. Acad. Sci. USA*, *93*, 13339–13344.

Seung, H. S., Lee, D. D., Reis, B. Y., & Tank, D. W. (2000). Stability of the memory of eye position in a recurrent network of conductance-based model neurons. *Neuron*, *26*, 259–271.

Stringer, S. M., Trappenberg, T. P., Rolls, E., & Aranjo, I. (2002). Self-organzing continuous attractor networks and path integration: One-dimensional models of head direction cells. *Network: Computation in Neural Systems*, *13*, 217–242.

Taube, J. S. (1998). Head direction cells and the neurophysiological basis for a sense of direction. *Prog. Neurobiol.*, *55*, 225–256.

Trappenberg, T. (1998). Dynamical cooperation and competition in a network of spiking neurons. In *Proc. Fifth International Conference on Neural and Intelligent Processing (ICONIP98)*. Kitakyushu, Japan.

Trappenberg, T. (2003). Continuous attractor neural networks. In L. N. de Castro & F. J. V. Zuben (Eds.), *Recent developments in biologically inspired computing*. Hershey, PA: Idee Group.

Tsodyks, M., & Sejnowski, T. (1995). Associative memory and hippocampal place cells. *Int. Journal of Neural Systems*, *6 (supp. 1995)*, 81–86.

van Vreeswijk C., & Somplinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, *274*, 1724–1726.

Wang, X. J. (2001). Synaptic reverberation underlying mnemonic persistent activity. *Trends in Neuroscience*, *24*, 455–463.

Wilson, M. A., & McNaughton, B. L. (1993). Dynamics of hippocampal ensemble code for space. *Science*, *261*, 1055–1058.

Wu, S., Amari, S., & Nakahara, H. (2002). Population coding and decoding in a neural field: A computational study. *Neural Computation*, *14*, 999–1026.

Wu, S., Chen, D., Niranjan, M., & Amari, S. (2003). Sequential Bayesian docoding with a population of neurons. *Neural Computation*, *15*, 993–1012.

Wu, S., Nakahara, H., & Amari, S. (2001). Population coding with correlation and an unfaithful model. *Neural Computation*, *13*, 775–797.

Zemel, R., Dayan, P., & Pouget, A. (1998). Probabilistic interpretation of population codes. *Neural Computation*, *10*, 403–430.

Zhang, K.-C. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory. *J. Neuroscience*, *16*, 2112–2126.

Zhang, K., Ginzburg, I., McNaughton, B., & Sejnowski, T. (1998). Interpreting neural population activity by reconstruction: Unified framework with application to hippocampal cells. *J. Neurophysiol.*, *79*, 1017–1044.

Zhang, K., & Sejnowski, T. J. (1999). Neural tuning: To sharpen or broaden. *Neural Computation*, *11*, 75–84.

Received August 23, 2004; accepted March 22, 2005.