Published in final edited form as:

IEEE Trans Biomed Eng. 2019 December ; 66(12): 3486–3498. doi:10.1109/TBME.2019.2906640.

Efficient Decoding of Multi-Dimensional Signals From Population Spiking Activity Using a Gaussian Mixture Particle Filter

Ali Yousefi
Department of Mathematics and Statistics, Boston University, Boston, MA 02215 USA.

Anna K. Gillespie, **Jennifer A. Guidera**
Kavli Institute for Fundamental Neuroscience and Department of Physiology, University of California.

Mattias Karlsson,
SpikeGadgets LLC.

Loren M. Frank,
Howard Hughes Medical Institute, Kavli Institute for Fundamental Neuroscience, and also with the Department of Physiology, University of California.

Uri T. Eden
Department of Mathematics and Statistics, Boston University.

Abstract

New recording technologies and the potential for closed-loop experiments have led to an increasing demand for computationally efficient and accurate algorithms to decode population spiking activity in multi-dimensional spaces. Exact point process filters can accurately decode low-dimensional signals, but are computationally intractable for high-dimensional signals. Approximate Gaussian filters are computationally efficient, but are inaccurate when the signals have complex distributions and nonlinear dynamics. Even particle filter methods tend to become inefficient and inaccurate when the filter distribution has multiple peaks. Here, we develop a new point process filter algorithm that combines the computational efficiency of approximate Gaussian methods with a numerical accuracy that exceeds standard particle filters. We use a mixture of Gaussian model for the posterior at each time step, allowing for an analytic solution to the computationally expensive filter integration step. During non-spike intervals, the filter needs only to update the mean, covariance, and mixture weight of each component. At spike times, a sampling procedure is used to update the filtering distribution and find the number of Gaussian mixture components necessary to maintain an accurate approximation. We illustrate the application of this algorithm to the problem of decoding a rat's position and velocity in a maze from hippocampal place cell data using both 2-D and 4-D decoders.

Corresponding author: Ali Yousefi, ayousefi@mgh.harvard.edu.

Index Terms—

Decoding; point process; place field; state space estimation; hippocampus

I. Introduction

ADVANCES in neural recording technology are making it possible to simultaneously record and manipulate the activity of large populations of cells [1], [2]. Numerous variables can influence this activity, and understanding how activity patterns represent information and contribute to specific computations requires analytical tools that are capable of extracting high dimensional information from these data [3], [4]. Having access to these patterns also makes it possible to use them in Brain-Machine Interfaces (BMIs) and to design and implement experimental interventions that can determine how specific patterns contribute to downstream activity and to behavior [5], [6]. BMIs and pattern-based interventions further require that the relevant multidimensional information be read out accurately and in real-time, but at present, general methods that enable real-time decoding of high dimensional structure from spike trains are not well developed.

Point process filtering has emerged as a powerful tool for estimating biological and behavioral signals from single and multiunit neural spiking data [7]–[11]. It has been used successfully to predict arm reaches from motor cortical ensembles [10], to identify aberrant rhythms in the basal ganglia of Parkinson's patients [11], and to decode the movement trajectories of rats from hippocampal place cells [8], [9], among many other applications. However, numerical computation of these filter algorithms is currently only tractable when the signals to estimate are very low dimensional. As a result, decoding using filter approaches is often done in a reduced 1D space, and decoding of 2D spaces is either very time consuming or uses more ad-hoc methods.

While these low dimensional approaches have been very useful, they cannot account for the high dimensional nature of the data. The spiking activity of single neurons typically relates to not one or two but many different covariates. As an example, the spiking of hippocampal place cells relates not only to animal's position in space [12], but also to its velocity [13], and past or intended future position [14], [15] among other variables. For higher dimensional data, approximate Gaussian [16], [17] and Sequential Monte Carlo (SMC) methods [18]–[20] are often used, but can suffer from substantial estimation bias and extraneous variability when the signals have non-Gaussian distributions or nonlinear dynamics. More recent techniques such as the ensemble Kalman filter (EnKF) [21] and Gaussian mixture filter [22]–[25] have been developed for non-linear and high-dimensional estimation problems; however, these methods do not fully address the issues discussed above [26]. The EnKF and Gaussian mixture filters assume additive normal noise in the observation process, limiting their applicability in the case in which the observation noise is not additive or normal. Furthermore, Gaussian mixture filters are built using a pre-set number of mixture components and lack a well-defined mechanism to control the growth or shrinkage of approximate Gaussian mixture components given the content of the observation process. These challenges led us to develop a new multidimensional point process filter procedure that takes advantage of the computational efficiency of Gaussian approximate methods but retains the accuracy of direct numerical computation of the filter distribution.

Two main advances contribute to the efficiency of this new point process filter algorithm. First, it uses a mixture of Gaussians approximation to the posterior filter distribution [23], [27], which allows for an analytic solution to the traditionally computationally expensive integration step of solving the filter equations. Second, a major improvement in efficiency is gained from treating spike and non-spike intervals distinctly. During non-spike intervals, the Gaussian mixture model approximation can remain accurate with updates only to the mean, covariance, and mixture weights of each component. In contrast, at spike times, we use a sampling procedure to update both the number of components in the Gaussian mixture model and the parameters for each component [28], [29]. Unlike in traditional particle filtering, this more computationally intensive sampling step only occurs at the fraction of time steps where spiking occurs.

To demonstrate the utility of this algorithm, we apply it to the decoding of a movement trajectory of a rat on a multi-arm track given the observed spiking activity of an ensemble of hippocampal place cells [30], [31]. We show the efficacy of the algorithm in 2 rats traversing the same track, each with an ensemble of greater than 50 hippocampal neurons. We perform the decoding using both a 2D position signal, which allows us to compare to the numerical solution point process filter equation, and using a 4D position and velocity signal, for which accurate numerical decoding would not be computationally feasible. The proposed algorithm can be applied to the general filter problem in high-dimensional spaces; it is specifically well suited to cases where the posterior distributions are multi-modal and show a complex structure.

The remainder of this paper is organized as follows: Section II describes the general formulation of the point process filter in multi-dimensional spaces given population spiking activity and develops the approximate Gaussian mixture particle filter solution. In Section III, we demonstrate the decoding result of the proposed filter solution in estimating a rat's movement trajectory during a memory-guided navigation task on a W-shaped maze. We also compare the performance and computational efficacy of the Gaussian mixture particle filter with the numerically computed filter solution and Gaussian approximation. In Section IV, we discuss the advantages and challenges of the proposed filter solution and possible future directions. The Appendix provides further discussion of place cells' receptive field properties, detailed derivation of gaussian approximation method, covariance matrix estimation, and methods to incorporate maze geometry constraints.

We use the following notation throughout this paper. Bold lowercase and uppercase letters are used to represent vectors and matrices, respectively. The state variable to be estimated is presented by $\boldsymbol{x_k}$, where k is time index. $\boldsymbol{x_k}$ consists of the position - (x_k, y_k) - and later includes the velocity - $(v_{x,k}, v_{y,k})$ – of the rat in the maze. Population spiking activity at time k is given by N_k set. The set of parameters -($\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$) - represents the mean vector and covariance matrix of a multivariate normal, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. $L(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the likelihood of observing sample $\boldsymbol{x}$ from a multivariate normal with $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ parameters; similarly, $L(\boldsymbol{x}; N_k)$ is the likelihood of $\boldsymbol{x}$ given current observation of population spiking activity. ∇ and ∇^2 are the gradient and Hessian operators.

II. Methods

A. Problem Definition

We model spiking observations as a point process using the conditional intensity, which defines the instantaneous probability of observing a spike at time t by

$$\lambda(t \mid H_t) = \lim_{\Delta t \to 0} Pr(\text{A spike in } (t, t + \Delta t] \mid H_t)/\Delta t \tag{1}$$

Here, H_t represents the full history of spiking from all recorded neurons up to time t. The probability of the neuron's firing a single spike in a small interval $[t, t + \Delta t)$ can be approximated as $\lambda(t|H_t)\Delta t$. This conditional intensity function is a history-dependent generalization of the inhomogeneous Poisson rate function [32]. We model neural spiking as a function of a covariate vector $\boldsymbol{x_t}$ by writing the intensity for each cell -$\lambda(t|H_t)$ - as a function of $\boldsymbol{x_t}$. These intensity models may be from a parametric class of models or based on non-parametric kernel estimates.

Under the point process framework, the instantaneous likelihood when at coordinate $\boldsymbol{x_k}$ of observing ΔN_k total spikes from an ensemble of C cells, with ΔN_k^1 spikes from cell 1, ΔN_k^2 spikes from cell 2, …, in the interval $\Delta_k = (t_{k-1}, t_k]$ is defined by

$$L(\boldsymbol{x_k}; N_k) \propto \begin{cases} \exp(-\Delta_k \Lambda(t_k \mid H_k)) & \Delta N_k = 0 \\ \prod_{c=1}^{C} [\lambda^c(t_k \mid H_k)\Delta_k]^{\Delta} N_k^c & \\ \exp(-\Delta_k \Lambda(t_k \mid H_k)) & \Delta N_k > 0 \end{cases} \tag{2}$$

$$N_k = \{\Delta N_k^c : c = 1 \cdots C\} \tag{3}$$

$$\Delta N_k = \sum_c \Delta N_k^c \tag{4}$$

where c refers to the cell index and $\lambda^c(t_k|H_k)$ represents the modeled intensity of cell c as a function of $\boldsymbol{x_k}$. In the definition of the likelihood function [9], [33], we assume the time interval is small enough that the likelihood of observing more than one spike per cell is negligible. We define the population intensity, $\Lambda(t_k|H_k)$ as the sum of the individual cells' conditional intensities.

$$\Lambda(t_k \mid H_k) = \sum_c \lambda^c(t_k \mid H_k) \tag{5}$$

We define the coordinate evolution $\boldsymbol{x_k}$ - also called the state process – as a Markov process, and the state evolution over time is given by a one-step density

$$\boldsymbol{x_k} | \boldsymbol{x_{k-1}} \sim f(\boldsymbol{x_k} \mid \boldsymbol{x_{k-1}}, \theta_{\boldsymbol{x}}) \tag{6}$$

where $\theta_{\boldsymbol{x}}$ is the model-free parameter. For example, the state evolution can be a linear model, and $\theta_{\boldsymbol{x}}$ might comprise the process mean and covariance matrices.

Given the observation process and state evolution equation, the exact posterior distribution of the state at time index k is defined by

$$p(\boldsymbol{x_k} \mid N_{1\ldots k}) \propto L(\boldsymbol{x_k}; N_k) \int p(\boldsymbol{x_k} \mid \boldsymbol{x_{k-1}}) \, p(\boldsymbol{x_{k-1}} \mid N_{1\ldots k-1}) \, \boldsymbol{d}\boldsymbol{x_{k-1}} \tag{7}$$

where the integral over $\boldsymbol{x_{k-1}}$ defines the one-step prediction in the Bayes filter paradigm [7]. The term $p(\boldsymbol{x_k} | N_{1\cdots k})$ is the filter estimate at time k given $\{N_1, \ldots, N_k\}$.

The one-step prediction stage of the filtering solution – the integral in equation (7) - presents a computational challenge. As there is no closed form solution of the one-step prediction for the point-process observation, a numerical approach is required. For a posterior distribution with a normal distribution and a linear state process – with a normal noise process, the one-step prediction has a closed form solution; however, for the point process observation, the integral requires to be calculated for each possible $\boldsymbol{x_k}$ and this calculation requires an integral over all possible values of $\boldsymbol{x_{k-1}}$. When $\boldsymbol{x_k}$ is multidimensional, the integral requires to be calculated for each point of multidimensional space $\boldsymbol{x_k}$ and it also becomes a multidimensional integral. Thus, the computational complexity of the numerical solution of the integral grows exponentially with the dimension. The computational cost will thus be of the order $O(n^{2d})$, where n is the number of samples over each axis of the state space and d is the state dimension. Note that the computation is updated at each time index, where these time steps are generally on the order of milliseconds. Even for a decoding problem in only two dimensions, with 1000 samples over each axis, the cost will be of the order $O(10^{12})$ per each time index. This rapidly becomes computationally impractical for real-time applications. For comparisons below, this strategy is referred to as the exact solution.

To build a computationally efficient solution, we discuss properties of the likelihood function of a point process observation given recordings of a neural ensemble. We then discuss how the one-step prediction and the filter solution can be efficiently approximated using a mixture model over time.

B. Approximate GMM Filter Solution

We assume that the posterior distribution of state at each time point – k – can be approximated by a Gaussian mixture model (GMM). The posterior of the state at time index $k - 1$ is defined by

$$p(\boldsymbol{x_{k-1}} \mid N_{1\cdots k-1}) \propto \sum_{s} \mathcal{N}(\boldsymbol{\mu_s}, \boldsymbol{\Sigma_s}) \pi_s \tag{8}$$

where π_s is the mixing weight and ($\boldsymbol{\mu_s}$, $\boldsymbol{\Sigma_s}$) are the mean vector and covariance matrix of the s^{th} mixture component. We assume there are S mixtures in total.

For simplicity, we begin with the assumption that the state evolution can be described by a linear state process

$$\boldsymbol{x_k} = \boldsymbol{A}\boldsymbol{x_{k-1}} + \boldsymbol{w_k} \quad \boldsymbol{w_k} \sim \mathcal{N}(0, \boldsymbol{Q}) \tag{9}$$

where $\boldsymbol{A}$ is the state matrix, $\boldsymbol{w_k}$ is the process noise, and $\boldsymbol{Q}$ is the process noise covariance matrix. Here, we assume that the elements of matrix $\boldsymbol{A}$ and covariance matrix $\boldsymbol{Q}$ are known. For a non-linear state transition process, we can use a multi-point linearization of the state-equation at each update time step [20, 25–p214]. Generally, the state trajectory follows a continuous and smooth path and its linearization gives an accurate approximation of the state evolution. Under this linear state transition process and the GMM approximation of the filter density from the previous time point, the one-step prediction density given by the integral on the right-hand side of equation (7) has an analytic solution, given by

$$p(\boldsymbol{x_k} \mid N_{1\ldots k-1}) \propto \sum_s \mathcal{N}(\boldsymbol{\mu_{os}}, \boldsymbol{\Sigma_{os}})\pi_{os} \tag{10}$$

where $\boldsymbol{\mu_{os}}$=$\boldsymbol{A\mu_s}$ is called the one-step prediction mean and $\boldsymbol{\Sigma_{os}} = \boldsymbol{A}\,\boldsymbol{\Sigma_s}\boldsymbol{A}' + \boldsymbol{Q}$ is called the one-step prediction covariance for the s^{th} mixture component. Under the linear state transition process, the mixing weights of one-step mixture components will be the same as the previous time point components, i.e., $\pi_{os} = \pi_s$. Notably, the GMM approximation substantially reduces the computational burden by eliminating the need to compute the integral in equation (7).

The next step in decoding is to update this one-step prediction density using the most recent observations from the neural population. Here we improve the efficiency of the update computation by separately considering time steps that include spiking and those that do not.

For any interval with no spiking observations, the likelihood function in equation (2) is defined by the population intensity, $\Lambda(t_k|H_k)$, and not by the individual intensities from any particular neuron – note that $\Delta N_k = 0$. Over these intervals, the filtering density diffuses slowly away from the local peaks of $\Lambda(t_k|H_k)$ [34], so that the difference between $p(\boldsymbol{x_k}|N_{1\cdots k})$ and $p(\boldsymbol{x_k}|N_{1\cdots k-1})$ is minor. For this reason, we choose not to update the number of components in the GMM model for all these intervals except long ones (described below), and instead only update the mean, covariance, and mixture weight of each component.

To compute the updates to the mean, covariance, and mixture weight, we multiply each mixture component of the GMM one-step density given by equation (10) by the likelihood of observing no new spiking. Next, we Taylor expand the logarithm of the likelihood about a different point for each mixture component; specifically, for the s^{th} mixture component, we expand the log likelihood about the one-step prediction mean for that component $\boldsymbol{\mu_{os}}$ using a second-order Taylor expansion [32]. Finally, we complete the square to generate new Gaussian mixtures for each component, yielding the following updates for the posterior mean, covariance, and mixture weight

$$\boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} \leftarrow \boldsymbol{\Sigma}_{\boldsymbol{os}}^{-1} + \nabla^2 \Lambda(\boldsymbol{\mu}_{\boldsymbol{os}})\Delta_k \tag{11}$$

$$\boldsymbol{\mu}_{\boldsymbol{s}} \leftarrow \boldsymbol{\mu}_{\boldsymbol{os}} - \boldsymbol{\Sigma}_{\boldsymbol{s}} \nabla \Lambda(\boldsymbol{\mu}_{\boldsymbol{os}})\Delta_k \tag{12}$$

$$\pi_s \leftarrow \pi_{os}\sqrt{\frac{\det \boldsymbol{\Sigma}_{\boldsymbol{s}}}{\det \boldsymbol{\Sigma}_{\boldsymbol{os}}}}\exp\left(-\Delta_k \Lambda(\boldsymbol{\mu}_{\boldsymbol{os}}) + 0.5\,\Delta_k^2 \nabla \Lambda(\boldsymbol{\mu}_{\boldsymbol{os}})^T \boldsymbol{\Sigma}_{\boldsymbol{s}} \nabla \Lambda(\boldsymbol{\mu}_{\boldsymbol{os}})\right) \tag{13}$$

where ∇ and ∇^2 are the gradient and Hessian of the log-likelihood function evaluated at the mean of one-step prediction mixture components. The gradient and Hessian can be calculated either numerically or analytically given how the conditional intensity is defined. To make the posterior estimate a probability distribution, we then normalize the sum of $\boldsymbol{\pi}_s$ s to one. Appendix A describes derivation of equations (11)–(13). Here, the mixture of Gaussians approximation starts by updating the component covariance matrices and then uses these updated values to update the component means and weights. We also check that each component's variance is positive definite, replacing any that are not with their one-step prediction values. In Appendix B, we propose a more robust Gaussian approximation method which guarantees the updated covariance matrices to be positive definite. In practice, when there is a long non-spiking period, we monitor the covariance of mixture components to avoid generating non-informative mixture components, for which the variance in each dimension significantly grows and the mixture becomes flattened over the space. Specifically, we check the mixtures' largest eigenvalues and trigger a spike-time procedure (described below) whenever this eigenvalue exceeds a predefined threshold.

Our update procedure for intervals that contain any spiking is fundamentally different, since spikes can cause substantial changes in the filter density and may require a different number of mixture components than the one-step prediction density. In this situation, we build a new GMM by drawing samples from a proposal distribution and approximating the samples' weights using a new GMM, which may have a different number of components.

The construction of the new GMM starts with a proposal distribution with the same number of components as the one-step prediction but with rescaled weights that reflect the likelihood of the most recent spiking activity [35], [36]. The new weight of the s^{th} component is defined by

$$\beta_s = L(\widehat{\boldsymbol{x}}_{\boldsymbol{s}}; N_k)\pi_{os} \tag{14}$$

where β_s is the un-normalized weight of the s^{th} component. The $\widehat{\boldsymbol{x}}_{\boldsymbol{s}}$ is defined by a function of the s^{th} component statistics; here, we set $\widehat{\boldsymbol{x}}_{\boldsymbol{s}}$ to be the mean of corresponding mixture component. The samples are drawn from this newly weighted proposal distribution, and the individual weights of each sample are defined by

$$w^p = p(\boldsymbol{x}^p \mid N_{1\ldots k}) / \sum_s \beta_s L(\boldsymbol{x}^p; \boldsymbol{\mu}_{\boldsymbol{os}}, \boldsymbol{\Sigma}_{\boldsymbol{os}}) \tag{15}$$

where $\boldsymbol{x}^p$ is the p^{th} sample from the proposal distribution, and w^p is the sample weight. We assume there are P samples in total; we then run the re-sampling step to derive P samples with equal weight. For the proposal distribution, we update the one-step mixture components' weights using the likelihood of the observed spiking. We could also adjust the mean and covariance of each mixture component in the proposal distribution, using the same update rule defined for non-spike time in equations (11)–(13). By incorporating the current observation in the proposal distribution, we tend to generate samples at locations where the posterior is larger, which potentially reduces the number of particles needed by an order of magnitude and avoids weight degeneracy [36].

Here, our focus is on solving the filter problem, assuming that the parameters for the observation and state models are known. When these parameters are unknown and need to be estimated along with the state process, we may augment these methods using recent techniques like SMC2 to dynamically update parameters of the observation and state processes [37]. Finally, in building our proposal distribution, we focus on improving the proposal density based on the one-step prediction density; for future offline applications, we could utilize methodologies like controlled SMC [38] which build proposal distributions that are optimal over the whole processing period.

Finally, we run an Expectation Maximization (EM) algorithm along with a Bayesian information criterion (BIC) to compute a new GMM with an updated number of components that parsimoniously approximates the posterior filtering distribution [39], [40]. To compute GMM, we need to know assignment of each particle to different mixture components; this information is not available and thus it becomes a latent variable in our GMM fitting problem. We thus utilize EM to iteratively estimate the particles' assignment to mixture components and update mixture components. We use BIC to control growth of the mixture components; without BIC penalty term, larger number of mixture components provide a better fit no matter of the fit significance. Under this EM algorithm, we run expectation (E-step) and maximization (M-step) steps recursively to update the parameters of a mixture given that the number of components is known. In the E-step, the expected assignment of each sample to the mixture components is evaluated. In the M-step, mixtures' parameters - means, covariances, and component weights - are recomputed given the expected assignment. The GMM estimation algorithm iterates between this EM procedure and a procedure to modify the number of mixture components as follows:

1. Run the EM to find a GMM with a conservatively large number of mixture components, $(\boldsymbol{\mu_s}, \boldsymbol{W_s}, \boldsymbol{\pi}_s)$ $s = 1 \cdots S$. The value of S can be a fixed large number, or it can be determined based on the number of neurons recorded or the number of peaks in the population intensity Λ over the state space.

2. Check all possible pairs of the GMM to find a pair with the lowest effect on the full likelihood. If we exclude the s_i and s_j components, we calculate the likelihood of remaining components by

$$f_{-i,j}(\boldsymbol{x}^p) = \frac{1}{\left(1 - \pi_{s_i} - \pi_{s_j}\right)} \sum_{s \notin \{s_i, s_j\}} \pi_s L(\boldsymbol{x}^p; \boldsymbol{\mu_s}, \boldsymbol{W_s}) \tag{16}$$

$$L_{-i,j} = \sum_p \log f_{-i,j}(\boldsymbol{x^p}) \tag{17}$$

We then pick the pair - (i, j) - that gives the highest value of $L_{-i,j}$.

3. Replace the removed pair with a new mixture component that maximizes the following cost function

$$\max_{\boldsymbol{\mu_e}, \boldsymbol{W_e}, \alpha} \sum_p \log\left((1-\alpha) f_{-i,j}(\boldsymbol{x^p}) + \alpha L(\boldsymbol{x^p}; \boldsymbol{\mu_e}, \boldsymbol{W_e})\right) \tag{18}$$

where, $(\boldsymbol{\mu_e}, \boldsymbol{W_e})$ are the mean vector and covariance matrix of the new mixture, which require estimation. $\alpha \in [0, 1]$ is the mixing weight between $f_{-i,j}(\cdot)$ and $L(\boldsymbol{x^p}; \boldsymbol{\mu_e}, \boldsymbol{W_e})$. The cost function is convex in α and the maximum occurs in the open interval (0, 1) for a known $(\boldsymbol{\mu_e}, \boldsymbol{W_e})$. The initial values of $(\boldsymbol{\mu_e}, \boldsymbol{W_e}, \alpha)$ are defined by

$$\boldsymbol{\mu_e} = \frac{\pi_i}{\pi_i + \pi_j}\boldsymbol{\mu_i} + \frac{\pi_j}{\pi_i + \pi_j}\boldsymbol{\mu_j} \tag{19.a}$$

$$\boldsymbol{W_e} = \frac{\pi_i}{\pi_i + \pi_j}\boldsymbol{W_i} + \frac{\pi_j}{\pi_i + \pi_j}\boldsymbol{W_j} + \frac{\pi_i \pi_j}{\left(\pi_i + \pi_j\right)^2}\left(\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\right)\left(\boldsymbol{\mu_i} - \boldsymbol{\mu_j}\right)^T \tag{19.b}$$

$$\alpha = \pi_i + \pi_j\,. \tag{19.c}$$

In Appendix C, we describe how these terms are estimated.

The weight and shape parameters of the new component are updated by an EM procedure to maximize the cost function with the following E- and M-steps.

E-step:

$$p(\mathrm{c_e} \mid \boldsymbol{x^p}) = \frac{\alpha \sum_p L(\boldsymbol{x^p}; \boldsymbol{\mu_e}, \boldsymbol{W_e})}{\sum_p \alpha L(\boldsymbol{x^p}; \boldsymbol{\mu_e}, \boldsymbol{W_e}) + (1-\alpha) f_{-i,j}(\boldsymbol{x^p})} \tag{20}$$

where $p(\mathrm{c_e}|\boldsymbol{x^p})$ is the expected assignment of $\boldsymbol{x^p}$ to the new mixture component, c_e.

M-step:

$$\boldsymbol{\mu_e} = \sum_p p\left(\mathrm{c_e} \mid \boldsymbol{x^P}\right)\boldsymbol{x^p} / \sum_p p(\mathrm{c_e} \mid \boldsymbol{x^p}) \tag{21}$$

$$\boldsymbol{W_e} = \sum_p p(\mathrm{c_e} \mid \boldsymbol{x^p})(\boldsymbol{x^p} - \boldsymbol{\mu_e})(\boldsymbol{x^p} - \boldsymbol{\mu_e})^T / \sum_p p(\mathrm{c_e} \mid \boldsymbol{x^p}) \tag{22}$$

$$\alpha = \sum_p p(c_e \mid \boldsymbol{x}^{\boldsymbol{p}}) \tag{23}$$

This EM procedure estimates the mean and covariance of only one mixture component plus its mixing weight; this suggests that numerical methods like gradient ascent can be utilized for simultaneous estimation of the $(\boldsymbol{\mu_e}, \boldsymbol{W_e}, \alpha)$ parameters.

4. Calculate the reduced model BIC and compare it with the original model. If the reduced model BIC is lower than the original one, replace the model with the updated one and go to step 2. Stop if the BIC criterion fails to reduce. The BIC for a GMM model with S mixture components - with $(\boldsymbol{\mu_s}, \boldsymbol{W_s}, \boldsymbol{\pi_s})s = 1\cdots S$ parameters - is equal to

$$BIC_S = -2\sum_p \log\left(\sum_s \pi_s L(\boldsymbol{x}^p; \boldsymbol{\mu_s}, \boldsymbol{W_s})\right) + \ln(P)\, D_g \tag{24.a}$$

$$D_g = S\left(\frac{d^2 - d}{2} + 2d + 1\right) - 1 \tag{24.b}$$

where, P is the number of samples and d is the dimension of $\boldsymbol{x}^{\boldsymbol{p}}$.

We perform steps 1 to 4 iteratively to find a minimum number of mixture components that properly approximate the posterior distribution of state at the spike time.

In this section, we developed an approximate GMM filter solution for both non-spiking and spike time periods. In Appendix D, we have summarized the processing steps of the complete solution. The resultant GMM comprises a minimum number of components which generally lie within a small spread and are evenly distributed over space. Through our analysis, we found these mixture components tend to better follow the state trajectory than do mixture components with a larger spread over space. Other choices of GMM might be examined depending on the problem definition.

III. Application to Decoding Spiking Activity of Place Cell Ensemble

We applied the GMM point process filter to neural data consisting of sorted spiking activity of ensembles of hippocampal place cells recorded from rats navigating a multi-arm track. Due to the close relationship between the firing of hippocampal place cells and spatial location of the animal in an environment, spike trains from multiple neurons can be used to decode a rat's location during behavior [30], [12]. Because the rat's position is also measured using video tracking software, we can assess decoding accuracy by comparing decoded position to the ground truth video-tracked position. For each of approximately 15 minute-long recording sessions, we use the first 85% of the session to build encoding models for each place cell, and we decode the remaining 15% of the recording session. For the 2D decoding, we compare performance metrics obtained with our GMM decoder to those obtained with the numerical computation of the exact solution filter, a standard particle filter, and a more traditional single Gaussian approximation. For the exact solution, we perform the computation using a coarse (2 cm) grid on both the x and y dimensions.

We also demonstrate 4D decoding, which additionally provides an estimate of the rat's velocity in the x and y direction. Because some place cells demonstrate directional firing - place fields that are specific to the rat traversing a region of track in a particular direction [40], we can also gain insight into the velocity of the animal by decoding the firing patterns. Using the velocity information, we are able to achieve greater accuracy in the decoded position compared to the 2D decoding. In the 4D case, we only report the performance of the GMM approximation solution, because estimating the exact solution in 4D is infeasible even for a coarse resolution in each axis.

A. 2-D Decoding

We use a non-parametric kernel method to build each cell's place field model in 2D space; $\boldsymbol{x_k} = (x_k, y_k)$ where x_k and y_k are the x and y coordinates at the time index k [41]. The conditional intensity per each cell is defined by

$$g(x_k, y_k \mid H_k) = \frac{\sum_{i=1}^{N} K\left(x_k - x_{ui}, y_k - y_{ui}; s_x, s_y\right)}{\Delta t \sum_{j=1}^{T} K\left(x_k - x_j, y_k - y_j; d_x, d_y\right)} \quad (25.a)$$

$$K\left(x, y; s_x, s_y\right) = \frac{1}{2\pi s_x s_y} \exp\left(-\frac{x^2}{2 s_x^2}\right) \exp\left(-\frac{y^2}{2 s_y^2}\right) \quad (25.b)$$

where x_j is the x position of the rat at time step j, y_j is the y position of the rat at time step j, and there are N spikes at times $0 < u_1 < \cdots < u_i < \cdots < u_N \leq T$ in the training session period, $(0, T]$. Note that we assume conditional intensity is stationary over the course of an experiment; this means we can use a pre-defined time window to estimate $g(x_k, y_k|H_k)$. Given this assumption, $g(x_k, y_k|H_k) \equiv g(x, y)$. K is a two-dimensional Gaussian-shaped kernel with a smoothness over space defined by the parameters (s_x, s_y) – s_x and s_y correspond to the kernel standard deviations in x and y directions. s_x and s_y represent a cell's place field and are set empirically given the cell response. The occupancy term, a measure of the time the rat spends at a specific coordinate, is defined by the same two-dimensional Gaussian kernel with a different smoothness term, (d_x, d_y). Δt is the observation update time, which is defined by the length T of the training session period divided by the number of observed samples in this period. In Appendix E, we show several examples of place cell spatial receptive fields as well as the likelihood function for non-spike intervals. Generally, d_x and d_y are set equal to or slightly larger than s_x, s_y to account for variability of the rat occupancy over the maze. Here, we set s_x and s_y to 6 and set d_x and d_y to 6. A more accurate estimation of these parameters can be attained by maximizing the likelihood of place cells activity using the $g(x_k, y_k|H_t)$ model, but this is beyond the scope of this research.

The state process evolution is defined by

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + \boldsymbol{q} \quad \boldsymbol{q} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix}\right) \tag{26}$$

where $\boldsymbol{q}$ is the process noise. The covariance matrix terms of $\boldsymbol{q}$ were estimated from the rat's movement statistics during the training session. Namely, they were set close to the average covariance of speed in the x and y directions during periods of rat mobility. The average $x-y$ covariance term was close to zero, and thus it is set to zero in the process noise covariance matrix in equation (26). The movement model defined here does not consider maze topology; in Appendix F, we discuss an addition to the decoder model that incorporates maze topology. Note that the 2-D random walk model proposed here provides minimal information about the exact movement trajectory of the rat; thus, we expect the observation process (the population spiking activity) provides enough information for the decoder to accurately estimate the rat movement trajectory. As previously mentioned, maze topology may also be factored into the observation process. However, building the exact model of the movement trajectory is complex, and is beyond the scope of this paper. We have defined the state model so that the observation process provides most of the information for decoding, and the state equation serves to constrain the movement to smooth trajectories. Note that although the state equation can be time varying, defining it as linear and state-independent allows us to build a more computationally efficient algorithm.

The approximate GMM, exact solution, particle filter, and Gaussian approximation were used to decode the position of a rat during the final 15% (2.2 minutes) of a 15-minute recording session. This testing dataset includes 4099 time points at a time resolution of 33 milliseconds ($\Delta t = 0.033$). The spatial resolution for the numerical solution of the exact filter in both the x and y dimensions is 2 cm; given the maze dimension, we have 50 samples in the x direction and 58 samples in the y direction. For the approximate GMM solution, we draw 4000 particles at each spike time and begin with 15 mixture components. The number of EM iterations for the first step of the EM algorithm is 250, and for the following steps is 50. For the Gaussian approximation, we use the same procedure with only a single component.

Table I shows the performance of different solutions in terms of the root mean squared error (RMSE) in cm, the percentage of time that the 95% highest posterior density (HPD) region [42] of the estimated filter density contains the rat's true position (95% HPD coverage), the average number of mixtures for the approximate GMM solution, and the 95% HPD area normalized by the 95% HPD area of the exact solution. Table I also shows the total runtime per each method; this time includes all necessary processing steps required to estimate the rat position posterior distribution for each processing timestep including conditional intensity of each cell using the non-parametric kernel – equation (25).

Figure 1 illustrates the decoding accuracy summary reported in Table I. At both time points, the Gaussian approximation tends to have a large covariance and provides a larger bias in estimating the rat's position. The approximate GMM estimate is close to the exact solution; the mixture components better capture the rat's actual position and the GMM estimate has a similar density to the exact solution. For this dataset, the RMSE for the (standard) particle filter is larger than GMM method, using the same number of particles. For the particle filter; the lower performance is likely due to the inability of this filter to capture multimodal distributions [36]. In Appendix D, we replicate the performance of these methods using data from an additional rat and multiple sessions.

B. 4-D Decoding

We again use a non-parametric kernel method similar to that used in equation (22) for 2D decoding to build each cell's conditional intensity in 4D given by $\boldsymbol{x_k} = (x_k, y_k, v_{x,k}, v_{y,k})$, where $(v_{x,k}, v_{y,k})$ represents the velocity of the rat's movement in the x and y directions respectively, computed using first differences of positions. The state equation is defined by

$$\begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ v_{x,k-1} \\ v_{y,k-1} \end{bmatrix} + \boldsymbol{q} \quad \boldsymbol{q} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 6 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 12 \end{bmatrix} \right) \tag{27}$$

where Δt is the update time step, and $\boldsymbol{q}$ is the process noise. For the covariance matrix terms of the process noise $\boldsymbol{q}$, we assume that the position and velocity state variables are uncorrelated. We use the same covariance terms for position as were used in the 2D problem, and we set the velocity covariance matrix terms based on the descriptive statistics of velocity during periods of movement in the training period.

The last row of Table I shows the decoding performance in 4D on the same data set used in the 2D decoding problem, also performed with 4000 particles. The decoding performance over position (x, y) surpasses that of the exact 2D solution, consistent with the notion that place cells also encode velocity.

Figure 2 shows the 4D decoder result at a single time point. Mixture components are centered around the rat's actual position (left) and velocity (right), consistent with the decoder accurately capturing position and velocity.

IV. Computational Efficiency

The computational complexity of numerical integration of the exact filter solution [9] in a d dimensional space through Riemann summation is on the order of $O(n^{2d})$, where n is the size of the partition in each dimension. The factor of 2 in the exponent comes from the one-step prediction density computation, which requires for each value of $\boldsymbol{x_k}$ integrating over all values of $\boldsymbol{x_{k-1}}$. This is the rate limiting step for complexity.

In contrast, for the Gaussian approximate filter [35], the computation of the one-step prediction distribution only requires matrix multiplication using a one step-prediction matrix that grows linearly with d. Conservatively, this has a computational complexity on the order of $O(d^3)$.

For the Gaussian mixture model approach described here, computation of the one-step prediction distribution still only involves matrix multiplication, but now for each mixture component separately. With *s* mixtures, this would be conservatively of order $O(sd^3)$. While *s* could depend on the dimensionality of the problem in principle, we are mostly only interested in posterior solutions that have a limited number of distinct peaks. For the GMM approach, the limiting step for computational complexity may come from the Monte Carlo estimation of the posterior at the spike times, rather than the one-step prediction computation. In that case, the computation would scale with the number of particles, *p*, used for estimation. Once again, *p* could scale with dimensionality, but if the posteriors in which we are interested have a limited number of peaks, we should be able to limit the number of particles used.

We also benefit from the closed form solution in non-spike time points in our proposed solution. The GMM method may excel when decoding with finer time resolutions so that the number of time steps that include spikes is relatively small.

The GMM method gives a parametric distribution at each time point. Using parametric distributions, the time required to complete post-processing computations – e.g., distribution statistics such as mean, variance, or mode – is significantly less than that of the exact solution. These calculations are embedded in the GMM computation cost, and as a result, the GMM method becomes a more suitable method for real-time applications.

A. Computational Cost for Our Example Decoders

One goal in developing more computationally efficient decoders for problems with multiple dimensions is to enable real-time estimation and closed-loop experiments that use the decoder output to influence stimulation. While we have yet to perform the optimization of code and computing resources to achieve real-time estimation, a simple comparison on our existing system suggests that this GMM can reduce computational cost substantially.

Here we present results from our current algorithms written in MATLAB 2016a and run on a Dual Core Intel i7 3.4 GHz processor with 64 GB RAM. We further used the MATLAB profiler to determine the computational time of different processing steps in the numerical computation of the 2D exact and GMM solutions. We focused on the computation of the one-step prediction density and the Monte Carlo estimation of the posterior, excluding the computation time for the point process intensities and likelihoods. We are concurrently working on the development of modeling methods to make these components more computationally efficient. Excluding the conditional intensity estimation, the exact solution with 2 cm resolution in *x* and *y* directions – 58×50 grid points – takes about 7.476 seconds to run per time point. The processing time using the GMM method using 2900 particles (a number equal to the number of grid points in the exact solution) for a spike interval is about 7.745 seconds. The GMM processing time for a non-spike time step is much faster: 0.186 seconds. With 1-msec time resolution, 93.2% of intervals are non-spike timesteps. While we must run the exact solution for every time point, we only need to run the full GMM method on 6.8% of data points. The average computational time is 682 msec per each time point using the GMM method compared to 7.476 seconds in the exact method. This is about 11 times faster than the exact solution.

These computational savings will scale up for higher dimension problems. For the 4D decoder, the GMM average computation time using 4000 particles is about 1.21 seconds, 0.285 seconds on non-spike times and 13.9 seconds on spike times, where the numerical exact decoder is computationally infeasible.

V. Discussion

Point process filtering has been successfully applied to a wide variety of neural data analysis problems, including decoding biological and behavioral signals from population spiking [8], [9], [12], tracking adaptation in neural coding properties [11], [32], and estimating parameters of biophysical neural models [9], [11], [33], [32]. With the development of new experimental methods and increasing interest in real-time and closed-loop experiments, the need for accurate and computationally efficient estimation algorithms from neural spiking data has grown tremendously. Here, we present a novel algorithm for solving the point process filter problem that combines computational benefits of approximate Gaussian methods with the potential accuracy in the face of multimodal filter distributions and nonlinear signal dynamics of exact numerical solutions.

The algorithm achieves these benefits by combining a Gaussian mixture model approximation to the filter distribution with an intermittent Monte Carlo sampling procedure that need only be conducted in intervals where the number of mixture components is likely to change substantially. Here, we chose to perform the resampling and update the number of mixtures only at times when spikes occurred because each spike can cause large changes in the filter density. Alternatively, we could have selected other periods for resampling; for example, we might resample only at periods where the spiking was unexpected given the current filter distribution, or we could be more conservative by resampling during some subset of non-spike times as well.

Our proposed algorithm improves the computational efficiency of the traditional point process filter procedure (the exact solution) in three ways. First, the GMM approach makes the integration step analytically solvable. Second, the update procedure at non-spike times has approximate analytic solutions that require simple computations. Finally, the more costly sampling procedure and computation of the appropriate number of mixture components occurs over only a fraction of time points. Though not explored here, these more costly steps are also readily parallelizable [43], allowing for the use of multicore computers or GPUs.

We applied the approximate GMM to decode the position (2D and 4D decoders) and velocity (4D decoder) of a rat based on the firing of hippocampal place cells. For 2D decoding, we compared performance metrics for decoded position using the approximate GMM with those obtained using the exact solution, standard particle filter, and the Gaussian approximation. Performance metrics were slightly better with the exact solution, but using more particles for approximate GMM may bridge that gap, and importantly the approximate GMM has the potential to be less computationally costly than the exact solution. The Gaussian approximation shows a larger RMSE and lower 95% HPD coverage that reflect bias and increased variability in the estimated posterior filter distribution. For 4D decoding, which involved estimation of position and velocity simultaneously, only the approximate

GMM was used, as 4D decoding would be computationally infeasible with the exact filter. The decoding accuracy in terms of RMSE is comparable to the exact solution in the 2D problem. The 95% HPD coverage in 4D decoding reaches 88%, which is close to the exact solution in 2D. Using a larger number of particles may further increase the HPD coverage of the 4D approximate GMM.

The algorithm we developed here is broadly applicable to the general point process filter problem in high-dimensional spaces. The algorithm may be particularly useful for processes that produce complex and multi-modal distributions. While we foresee this approach being useful for a number of applications, one immediate direction we are pursuing is its use in developing closed-loop experiments. Particularly, we are interested in decoding replay events which will be used to stimulate a hippocampal population to better understand the role of replay in rule learning and memory formation.

There are several modeling challenges which might be investigated in future research. Methods for optimally selecting the initial number of mixture components in the EM algorithm would be helpful. Algorithms that can modify the number of mixture components in batch rather than sequentially would also improve performance. We demonstrated our approach using a linear state equation process; extending the idea developed here for non-linear state processes would be important for task with complex state dynamics. Further, extending this algorithm to datasets with more state variables (such as decoding during an arm reaching task) may highlight the pros and cons of this decoding framework.

VI. Conclusion

In this research, we proposed a computationally efficient point-process filter solution for multi-dimensional spaces. The methodology has been applied to decode a rat movement trajectory in a W-shaped maze. For the 2D decoding, we estimate the rat position given an ensemble of spiking activity. For the 4D decoding, we estimate the rat position and movement velocity given the same ensemble of spiking activity. In the 2D problem, we compared the performance of the proposed approximate GMM filter with that of an exact filter, standard particle filter, and Gaussian approximation in 2D. The approximate GMM solution shows a similar performance to the exact solution, whereas its computational cost is significantly lower than the exact solution. The Gaussian approximation shows a large RMSE and lower 95% HPD coverage that suggest a poor estimation of the posterior distribution. For the 4D decoding problem, the performance on position trajectory is close to the exact solution in the 2D problem. The 95% HPD coverage in 4D decoding reaches 87.8%, which is close to the exact solution even with a small number of particles used in this analysis. We utilized GMM to approximate multi-modal distributions and proposed a computationally efficient update rule to estimate the filter solution on different observation times. In approximating the posterior distribution, we used a hybrid update rule given different dynamics of the observed signal. We used a modified Gaussian approximation on the non-spiking time points, and a Monte Carlo method combined with revised GMM estimation procedure on the spike time points; using this hybrid methodology, we are able to build a fast decoder algorithm without losing accuracy. The proposed methodology is applicable to the general filter problem in high-dimensional spaces, specifically for observation processes showing a complex and multi-modal distribution.

Acknowledgments

This work was supported by R01 MH105174 and SCGB Grant 320135.

Appendix

A. Gaussian Approximation on a Non-Spike Time

For the s^{th} mixture component, we require to approximate the following term with a new mixture component

$$\pi_s L(\boldsymbol{x_k}; \boldsymbol{\mu_s}, \boldsymbol{\Sigma_s}) \cong \pi_{os} L(\boldsymbol{x_k}; \boldsymbol{\mu_{os}}, \boldsymbol{\Sigma_{os}}) \exp(-\Lambda(\boldsymbol{x_k})\Delta_k) \tag{A.1}$$

We first start by taking logarithm of both sides of (A.1).

$$\begin{aligned} &-\frac{1}{2}(\boldsymbol{x_k} - \boldsymbol{\mu_s})^T \boldsymbol{\Sigma_s}^{-1}(\boldsymbol{x_k} - \boldsymbol{\mu_s}) \\ &\cong -\frac{1}{2}(\boldsymbol{x_k} - \boldsymbol{\mu_{os}})^T \boldsymbol{\Sigma_{os}^{-1}}(\boldsymbol{x_k} - \boldsymbol{\mu_{os}}) - \Lambda(\boldsymbol{x_k})\Delta_k + E \end{aligned} \tag{A.2}$$

where, E consists of all other terms not including $\boldsymbol{x_k}$. To find the update rule for the mixture mean and covariance, we take the first and second derivative of both sides of (A.2) with respect to $\boldsymbol{x_k}$. The first derivative is defined by

$$-\boldsymbol{\Sigma_s}^{-1}(\boldsymbol{x_k} - \boldsymbol{\mu_s}) = -\boldsymbol{\Sigma_{os}^{-1}}(\boldsymbol{x_k} - \boldsymbol{\mu_{os}}) - \nabla\Lambda(\boldsymbol{x_k})\Delta_k \tag{A.3}$$

and the second derivative is defined by

$$\boldsymbol{\Sigma_s^{-1}} = \boldsymbol{\Sigma_{os}^{-1}} + \nabla^2\Lambda(\boldsymbol{x_k})\Delta_k \tag{A.4}$$

We assume that (A.1) and (A.2) are valid for any values of $\boldsymbol{x_k}$; thus, we set $\boldsymbol{x_k}$ to $\boldsymbol{\mu_{os}}$ [32]. By setting $\boldsymbol{x_k}$ to $\boldsymbol{\mu_{os}}$, we can get equations (11) and (12). Here, we need to update the covariance matrix first and then the mean vector. The other possible solution is setting $\boldsymbol{x_k}$ to $\boldsymbol{\mu_s}$; under this assumption, we get another solution where the update rule starts by updating the mean and then covariance matrix.

We also need to estimate updated mixing weight, $\boldsymbol{\pi_s}$. We assume that the likelihood of the new normal distribution should be the same of the right over possible $\boldsymbol{x_k}$ including $\boldsymbol{\mu_{os}}$; though it can be estimated on any other point, $\boldsymbol{\mu_{os}}$ represents the most probable point of the one-step prediction and thus update rule for mixing weight - $\boldsymbol{\pi_s}$ - is defined by sett x_k to $\boldsymbol{\mu_{os}}$

$$\begin{aligned} &\pi_s \sqrt{\det(2\pi\boldsymbol{\Sigma}_{\mathrm{s}})} \exp\left(-\frac{1}{2}\Delta_k^2 \nabla\Lambda \boldsymbol{\Sigma}_{\mathrm{s}} \nabla\Lambda(\boldsymbol{x_k})\right) \\ &= \pi_{os}\sqrt{\det(2\pi\boldsymbol{\Sigma_{os}})} \exp(-\Lambda(\boldsymbol{\mu_{os}})\Delta_k) \end{aligned} \tag{A.5}$$

where, the left side is calculated using the multivariate normal with mean and covariance of $\boldsymbol{\mu_s}$ and $\boldsymbol{\Sigma_s}$. The update rule for mixing weight defined in equation (13) is derived by (A.5).

B. Revised Gaussian Approximation With Guaranteed Positive Definite Covariance

When Gaussian approximation is used during non-spike time periods to update the posterior distribution, it is important to ensure that the covariance estimate of posterior mixture components remains positive-definite. The conditional intensity estimation and likelihood function are multi-modal, which may cause the posterior covariance estimation of some of the mixture components to be non-negative. One solution is to avoid updating these mixtures, but a better solution is to control eigenvalues of these covariance matrices to ensure all are positive.

Here, we will discuss the idea for one mixture component, and its extension for multiple mixtures is trivial. The objective is to approximate the posterior using a multivariate Gaussian with a guaranteed positive-definite covariance matrix. Logarithm of the posterior is defined by

$$\log \boldsymbol{x_k} \mid \boldsymbol{N}_{1 \dots \boldsymbol{k}} \propto \log \boldsymbol{x_k} \mid \boldsymbol{N}_{1 \dots \boldsymbol{k}-1} + \log \exp(-\Lambda_k \Delta t) \tag{B.1}$$

Now, we use Taylor expansion around $\boldsymbol{m_{k|k-1}}$ to approximate the posterior distribution with a Gaussian distribution

$$\begin{aligned} &\log \boldsymbol{x_k} \mid \boldsymbol{N}_{1 \dots \boldsymbol{k}} \propto \\ &-\frac{1}{2}\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right)^T \boldsymbol{\Sigma}_{\boldsymbol{k} \mid \boldsymbol{k}-1}^{-1}\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right) \cdots \\ &- \nabla \Lambda_k\left(\boldsymbol{m_{k \mid k-1}}\right)\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right)\Delta t \cdots \\ &-\frac{1}{2}\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right)^{\mathrm{T}} \nabla^2 \Lambda_k\left(\boldsymbol{m_{k \mid k-1}}\right) \\ &\times\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right)\Delta t + \mathrm{C} \end{aligned} \tag{B.2}$$

where ∇ and ∇^2 are the gradient and Hessian of the conditional intensity at $\boldsymbol{m_{k|k-1}}$. Variable C is the remainder of the Taylor series.

To make sure that the posterior covariance is positive-definite, we use the following approximation.

$$\begin{aligned} &\log\ \boldsymbol{x_k} \mid \boldsymbol{N}_{1 \dots \boldsymbol{k}} \propto \\ &-\frac{1}{2}\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right)^T\left(\boldsymbol{\Sigma}_{\boldsymbol{k} \mid \boldsymbol{k}-1}^{-1} + r\,\nabla^2 \Lambda_k\left(\boldsymbol{m_{k \mid k-1}}\right)\Delta t\right) \\ &\times\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right) - \nabla \Lambda_k\left(\boldsymbol{m_{k \mid k-1}}\right)\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right)\Delta t \dots \\ &-\frac{1}{2}(1-r)\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right)^T \nabla^2 \Lambda_k\left(\boldsymbol{m_{k \mid k-1}}\right) \\ &\times\left(\boldsymbol{x_k} - \boldsymbol{m_{k \mid k-1}}\right)\Delta t + \mathrm{C} \end{aligned} \tag{B.3}$$

Equation (B.3) can be rewritten as

$$\begin{aligned}\log \boldsymbol{x}_{\boldsymbol{k} \mid \boldsymbol{N}_{1 \ldots \boldsymbol{k}}} \propto & -\frac{1}{2}\left(\boldsymbol{x}_{\boldsymbol{k}}-\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right)^{T} \\ & \times\left(\boldsymbol{\Sigma}_{\boldsymbol{k} \mid \boldsymbol{k}-1}^{-1}+r \nabla^{2} \Lambda_{k}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right) \Delta t\right)\left(\boldsymbol{x}_{\boldsymbol{k}}-\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right) \\ & -\left(\nabla \Lambda_{k}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right)+\frac{1}{2}(1-r)\left(\boldsymbol{x}_{\boldsymbol{k}}-\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right)^{\mathrm{T}}\right. \\ & \left.\times \nabla^{2} \Lambda_{k}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right)\right) \\ & \times\left(\boldsymbol{x}_{\boldsymbol{k}}-\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right) \Delta t+C\end{aligned} \tag{B.4}$$

The update rule for the posterior covariance matrix is defined by (11) – we assume the posterior is multivariate normal with mean $\boldsymbol{m}_{\boldsymbol{k}|\boldsymbol{k}}$ and covariance $\Sigma_{\boldsymbol{k}|\boldsymbol{k}}$. Here, we work to rewrite the right side of equation (A.4) using a multivariate normal distribution with mean $\boldsymbol{m}_{\boldsymbol{k}|\boldsymbol{k}}$ and covariance $\Sigma_{\boldsymbol{k}|\boldsymbol{k}}$.

$$\boldsymbol{\Sigma}_{\boldsymbol{k} \mid \boldsymbol{k}}^{-1}=\boldsymbol{\Sigma}_{\boldsymbol{k} \mid \boldsymbol{k}-1}^{-1}+r \nabla^{2} \Lambda_{k}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right) \Delta t \tag{B.5}$$

where we can find the largest r – $0 < r < 1$ – that keeps the updated covariance positive definite. We check the eigenvalues of $\Sigma_{\boldsymbol{k}|\boldsymbol{k}}$, and select the largest value of r when all the corresponding eigenvalues are positive or larger than a minimum threshold. Given the posterior covariance, we build the update rule for the posterior mean by

$$\begin{aligned}\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}}= & \boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}-\boldsymbol{\Sigma}_{\boldsymbol{k} \mid \boldsymbol{k}}\left(\nabla \Lambda_{k}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right)+\frac{1}{2}(1-\mathrm{r})\right. \\ & \left.\times \nabla^{2} \Lambda_{k}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right)\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}}-\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right)\right) \Delta t\end{aligned} \tag{B.6}$$

Solving equation (B.6) gives

$$\begin{aligned} & \boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}}\left(I+\frac{1}{2}(1-\mathrm{r}) \boldsymbol{\Sigma}_{\boldsymbol{k} \mid \boldsymbol{k}} \nabla^{2} \Lambda_{\mathrm{k}}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right) \Delta t\right) \\ & =\cdots \boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}-\boldsymbol{\Sigma}_{\boldsymbol{k} \mid \boldsymbol{k}} \\ & \times\left(\nabla \Lambda_{k}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right)-\frac{1}{2}(1-r) \nabla^{2} \Lambda_{k}\left(\boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right) \boldsymbol{m}_{\boldsymbol{k} \mid \boldsymbol{k}-1}\right) \Delta t\end{aligned} \tag{B.7}$$

Thus, we first run equation (B.5) to estimate the covariance matrix and then run equation (B.7) to update the posterior mean. Note that by setting r to zero, the posterior covariance becomes equal to the one-step covariance matrix. On the other end, an r equal to one causes a smoother change for the likelihood function around $\boldsymbol{m}_{\boldsymbol{k}|\boldsymbol{k}-1}$, which is the usual behavior when the Gaussian approximation is utilized.

C. Mean and Variance Update Rule

Here, we want to replace the two mixture components ($\boldsymbol{\mu}_{\boldsymbol{i}}$, $\boldsymbol{W}_{\mathbf{i}}$, π_{i}) and ($\boldsymbol{\mu}_{\boldsymbol{i}}$, $\boldsymbol{W}_{\mathbf{i}}$, π_{j}) with a single mixture component with mean and covariance parameters equal to those derived by a mixture model constructed using these two mixture components. These two mixture components are a subset of larger set of mixture components and thus the sum of π_{i} and π_{j} – their mixing weights - is not necessarily one. α which is the sum of π_{i} and π_{j} represents the weight of the new mixture component. We first build a new GMM consisting of these two mixtures with normalized mixing weights of $\frac{\pi_i}{\pi_i + \pi_j}$ and $\frac{\pi_\mathrm{j}}{\pi_\mathrm{i} + \pi_\mathrm{j}}$. This new GMM distribution – represent by X_e random variable, is defined by

$$X_e \propto \frac{\pi_\mathrm{i}}{\pi_\mathrm{i} + \pi_\mathrm{j}} \mathcal{N}(\boldsymbol{\mu_i}, \boldsymbol{\Sigma}_\mathrm{i}) + \frac{\pi_\mathrm{j}}{\pi_\mathrm{i} + \pi_\mathrm{j}} \mathcal{N}(\boldsymbol{\mu_j}, \boldsymbol{\Sigma}_\mathrm{j}) \tag{C.1}$$

Using this distribution, we can define the mean and covariance of X_e. Mean of X_e is defined by

$$\boldsymbol{\mu_e} = \mathrm{E}[\mathrm{X_e}] = \frac{\pi_\mathrm{i}}{\pi_\mathrm{i} + \pi_\mathrm{j}} \boldsymbol{\mu_i} + \frac{\pi_\mathrm{j}}{\pi_\mathrm{i} + \pi_\mathrm{j}} \boldsymbol{\mu_j} \tag{C.2}$$

which is the mean over space given by X_e distribution. X_e covariance is defined by

$$\begin{aligned} \boldsymbol{\Sigma_e} &= \mathrm{E}\left[\mathrm{X_e} X_e^T\right] - \boldsymbol{\mu_e}\boldsymbol{\mu_e^T} = \frac{\pi_\mathrm{i}}{\pi_\mathrm{i} + \pi_\mathrm{j}}\left(\boldsymbol{\Sigma_i} + \boldsymbol{\mu_i}\boldsymbol{\mu_i^T}\right) \\ &+ \frac{\pi_\mathrm{j}}{\pi_\mathrm{i} + \pi_\mathrm{j}}\left(\boldsymbol{\Sigma_j} + \boldsymbol{\mu_j}\boldsymbol{\mu_j^T}\right) - \boldsymbol{\mu_e}\boldsymbol{\mu_e^T} \end{aligned} \tag{C.3}$$

where, by replacing μ_e from (C.1) - or (19.a), we get $\boldsymbol{\Sigma}_\mathbf{e}$ defined in (19.b).

Now, $\boldsymbol{\mu_e}$ and $\boldsymbol{\Sigma}_\mathbf{e}$ define the mean and covariance of the new mixture component. This component weight - α - is the sum of π_i and π_j.

D. Complete Solution Processing Steps

Table II provides pseudo-code detailing each processing step of the filter solution on both spike and non-spike time points. In Table II, other parameters of the solution are reset to their initial value at a new time step.

E. Place Cell Receptive Fields

Figure 3 shows several examples of place cell spiking patterns. For each cell, we display the location of the rat at the time of each spike as well as the smoothed estimates of the contribution to the spatial likelihood in 2D at a spike time for that neuron. As established in prior literature, place cells' receptive fields have diverse locations, extents, and topologies, and are often not unimodal [13], [14], [44]. This suggests that the posterior estimate of the position given these cells' spiking activity can also have a multi-modal distribution [45].

Figure 4 shows the likelihood function over space on non-spike intervals. The likelihood is non-zero over the entire maze and becomes relatively flat when there are many place cells covering the maze area. Note that we have less than 100 putative cells in our experiments, out of which a limited number of cells show consistent neural activity and distinct receptive field activity. When there are many place cells, the likelihood on non-spike times becomes flat, suggesting that the posterior estimate simply broadens at non-spike times.

F. Including Maze Information in the Decoder

We modeled the rat's movement as a random walk over 2D space, and thus the movement model does not take into account maze topology. However, in reality, the rat's position is constrained to the track area. Although it is possible to incorporate maze boundaries into the movement model, we prefer to include minimal information about it in our encoder model. This allows us to see how much information is embedded in the place cells spiking activity, and how well our decoder can trace the rat's movement with minimal assumptions about its movement pattern. However, we imposed the maze topology in the likelihood function. To do this, we define a penalty term which becomes a small number ($\ll 1$) for any point outside the maze, and equal to one for any point inside the maze. The likelihood function is then multiplied by this value. This modified likelihood function is defined by eq. (F1), (F2) shown at the bottom of this page.

$$\widehat{L}(\boldsymbol{x_k}; N_k) \propto L(\boldsymbol{x_k}; N_k)\, g(\boldsymbol{x_k}) \tag{F.1}$$

$$g(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x} \in \{\boldsymbol{x_k} : \boldsymbol{x_k} \text{ are the coordinate of points inside the maze}\} \\ \varepsilon & \boldsymbol{x} \notin \{\boldsymbol{x_k} : \boldsymbol{x_k} \text{ are the coordinate of points inside the maze}\} \end{cases} \tag{F.2}$$

where g($\boldsymbol{x}$) is the penalty term. The penalty term pushes the posterior estimate to the area inside the maze, and as result reflects the maze topology.

Figure 5 shows the penalty area, which includes all coordinates farther than a pre-defined minimum distance (3.5 cm) from the training trajectory points. The penalty term was set to 10^{-6} for both 2D and 4D decoders. The term only carries information about the maze topology, not velocity constraints.

G. Performance Results in Multiple Datasets

Table III shows the decoding results for neural data from two rats over four experimental sessions. For each rat, we assessed the decoding algorithm on two separate recording sessions, which occurred on different days and may include distinct neural populations. The results for rat 2, session 1 are the ones reported in Table I. The performance results for the additional rat and sessions are consistent with the general findings presented in the main text.

H. Source Code and Sample Data

A copy of the source code written to implement the decoding methodologies presented in this paper are available at the GitHub repository https://github.com/Eden-Kramer-Lab/Multi-Dimensional-Decoder. The repository also includes a copy of the data analyzed in this paper. Beside the decoding result, the code provides performance metrics of different methods. The source code also provides options to change number of particles in the particle-filtering and GMM method, and percentage of training and testing dataset. In GitHub repository, we also included a movie, which shows the decoding result for a complete session of the experiment.

References

[1]. Buzsáki G, “Large-scale recording of neuronal ensembles,” Nature Neurosci, vol. 7, no. 5, pp. 446–451, 2004. [PubMed: 15114356]

[2]. Michon F et al., “Integration of silicon-based neural probes and micro-drive arrays for chronic recording of large populations of neurons in behaving animals,” J. Neural Eng, vol. 13, no. 4, p. 046018, 2016. [PubMed: 27351591]

[3]. Grosmark AD and Buzsáki G, “Diversity in neural firing dynamics supports both rigid and learned hippocampal sequences,” Science, vol. 351, no. 6280, pp. 1440–1443, 2016. [PubMed: 27013730]

[4]. Chen Z and Wilson MA, “Deciphering neural codes of memory during sleep,” Trends Neurosci, vol. 40, pp. 260–275, 2017. [PubMed: 28390699]

[5]. Ego-Stengel V and Wilson MA, “Disruption of ripple-associated hippocampal activity during rest impairs spatial learning in the rat,” Hippocampus, vol. 20, no. 1, pp. 1–10, 2010. [PubMed: 19816984]

[6]. Jadhav SP et al., “Awake hippocampal sharp-wave ripples support spatial memory,” Science, vol. 336, no. 6087, pp. 1454–1458, 2012. [PubMed: 22555434]

[7]. Koyama S et al., “Bayesian decoding of neural spike trains,” Ann. Inst. Statistical Math, vol. 62, no. 1, pp. 37–59, 2010.

[8]. Huang Y et al., “Decoding movement trajectories through a T-maze using point process filters applied to place field data from rat hippocampal region CA1,” Neural Comput, vol. 21, no. 12, pp. 3305–3334, 2009. [PubMed: 19764871]

[9]. Brown EN et al., “A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells,” J. Neurosci, vol. 18, no. 18, pp. 7411–7425, 1998. [PubMed: 9736661]

[10]. Brockwell AE et al., “Recursive Bayesian decoding of motor cortical signals by particle filtering,” J. Neurophysiol, vol. 91, no. 4, pp. 1899–1907, 2004. [PubMed: 15010499]

[11]. Sarma SV et al., “Using point process models to compare neural spiking activity in the subthalamic nucleus of Parkinson’s patients and a healthy primate,” IEEE Trans. Biomed. Eng, vol. 57, no. 6, pp. 1297–1305, 6 2010. [PubMed: 20172804]

[12]. O’Keefe J and Dostrovsky J, “The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely moving rat,” Brain Res, vol. 34, pp. 171–175, 1971. [PubMed: 5124915]

[13]. McNaughton BL et al., “The contributions of position, direction, and velocity to single unit activity in the hippocampus of freely-moving rats,” Exp. Brain Res, vol. 52, no. 1, pp. 41–49, 1983. [PubMed: 6628596]

[14]. Frank LM et al., “Trajectory encoding in the hippocampus and entorhinal cortex,” Neuron, vol. 27, no. 1, pp. 169–178, 2000. [PubMed: 10939340]

[15]. Wood ER et al., “Hippocampal neurons encode information about different types of memory episodes occurring in the same location,” Neuron, vol. 27, no. 3, pp. 623–633, 2000. [PubMed: 11055443]

[16]. Julier SJ and Uhlmann JK, “New extension of the Kalman filter to nonlinear systems,” in Proc. AeroSense Int. Soc. Opt. Photon, 7 1997, pp. 182–193.

[17]. Smith AC and Brown EN, “Estimating a state-space model from point process observations,” Neural Comput, vol. 15, no. 5, pp. 965–991, 2003. [PubMed: 12803953]

[18]. Arulampalam MS et al., “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” IEEE Trans. Signal Process, vol. 50, no. 2, pp. 174–188, 2 2002.

[19]. Ergun A et al., “Construction of point process adaptive filter algorithms for neural systems using sequential Monte Carlo methods,” IEEE Trans. Biomed. Eng, vol. 54, no. 3, pp. 419–428, 3 2007. [PubMed: 17355053]

[20]. Djuric PM et al., “Particle filtering,” IEEE Signal Process. Mag, vol. 20, no. 5, pp. 19–38, 9 2003.

[21]. Bengtsson T et al., “Toward a nonlinear ensemble filter for high-dimensional systems,” J. Geophys. Res.: Atmos, vol. 108, no. D24, 2003.

[22]. Hoteit I et al., “A new approximate solution of the optimal nonlinear filter for data assimilation in meteorology and oceanography,” Monthly Weather Rev, vol. 136, no. 1, pp. 317–334, 2008.

[23]. Kotecha JH and Djuric PM, “Gaussian sum particle filtering,” IEEE Trans. Signal Process, vol. 51, no. 10, pp. 2602–2612, 10 2003.

[24]. Chen R and Liu JS, “Mixture Kalman filters,” J. Roy. Statistical Soc.: Ser. B (Statistical Methodol.), vol. 62, no. 3, pp. 493–508, 2000.

[25]. Anderson BD and Moore JB. Optimal Filtering, vol. 21 Englewood Cliffs, NJ, USA: Dover, 1979, pp. 22–95.

[26]. Stordal AS et al., “Bridging the ensemble Kalman filter and particle filters: The adaptive Gaussian mixture filter,” Comput. Geosci, vol. 15, no. 2, pp. 293–305, 2011.

[27]. Reynolds D, “Gaussian mixture models,” in Encyclopedia of Biometrics. Li SZ and Jain A, eds. Boston, MA: Springer US, 2009, pp. 659–663.

[28]. Celeux G and Soromenho G, “An entropy criterion for assessing the number of clusters in a mixture model,” J. Classification, vol. 13, no. 2, pp. 195–212, 1996.

[29]. Verbeek JJ et al., “Efficient greedy learning of Gaussian mixture models,” Neural Comput, vol. 15, no. 2, pp. 469–485, 2003. [PubMed: 12590816]

[30]. Moser EI et al., “Place cells, grid cells, and the brain’s spatial representation system,” Annu. Rev. Neurosci, vol. 31, pp. 69–89, 2008. [PubMed: 18284371]

[31]. Lu X and Bilkey DK, “The velocity-related firing property of hippocampal place cells is dependent on self-movement,” Hippocampus, vol. 20, no. 5, pp. 573–583, 2010. [PubMed: 19554643]

[32]. Eden UT et al., “Dynamic analysis of neural encoding by point process adaptive filtering,” Neural Comput, vol. 16, no. 5, pp. 971–998, 2004. [PubMed: 15070506]

[33]. Paninski L, “Maximum likelihood estimation of cascade point-process neural encoding models,” Netw.: Comput. Neural Syst, vol. 15, no. 4, pp. 243–262, 2004.

[34]. Eden UT and Brown EN, “Continuous-time filters for state estimation from point process models of neural data,” Statistica Sinica, vol. 18, no. 4, pp. 1293–1310, 2008. [PubMed: 22065511]

[35]. Pitt MK and Shephard N, “Filtering via simulation: Auxiliary particle filters,” J. Amer. Statistical Assoc, vol. 94, no. 446, pp. 590–599, 1999.

[36]. Snyder C, “Particle filters, the “optimal” proposal and high-dimensional systems,” in Proc. ECMWF Seminar Data Assimilation Atmos. Ocean, 9 2011, pp. 1–10.

[37]. Chopin N et al., “SMC2: An efficient algorithm for sequential analysis of state space models,” J. Roy. Statistical Soc.: Ser. B (Statistical Methodology), vol. 75, no. 3, pp. 397–426, 2013.

[38]. Heng J et al., “Controlled sequential Monte Carlo,” 2017, arXiv:1708.08396

[39]. Gardiner CW, Handbook of stochastic methods, vol. 3 Springer, Berlin, 1985.

[40]. Dempster AP, Laird NM, and Rubin DB, “Maximum likelihood from incomplete data via the EM algorithm,” J. R. Stat. Soc. Ser. B, vol. 39, no. 1, pp. 1–22, 1977.

[41]. Deng X et al., “Clusterless decoding of position from multiunit activity using a marked point process filter,” Neural Comput, vol. 27, pp. 1438–1460, 2015. [PubMed: 25973549]

[42]. Hyndman RJ, “Computing and graphing highest density regions,” Amer. Statistician, vol. 50, no. 2, pp. 120–126, 1996.

[43]. Kumar NP et al., “Fast parallel expectation maximization for Gaussian mixture models on GPUs using CUDA,” in Proc. 11th IEEE Int. Conf. High Performance Comput. Commun, 6 2009, pp. 103–109.

[44]. Fenton AA et al., “Unmasking the CA1 ensemble place code by exposures to small and large environments: More place cells and multiple, irregularly arranged, and expanded place fields in the larger space,” J. Neurosci, vol. 28, no. 44, pp. 11250–11262, 2008. [PubMed: 18971467]

[45]. Agarwal R et al., “A novel nonparametric approach for neural encoding and decoding models of multimodal receptive fields,” Neural Comput, vol. 28, no. 7, pp. 1356–1387, 2016. [PubMed: 27172447]

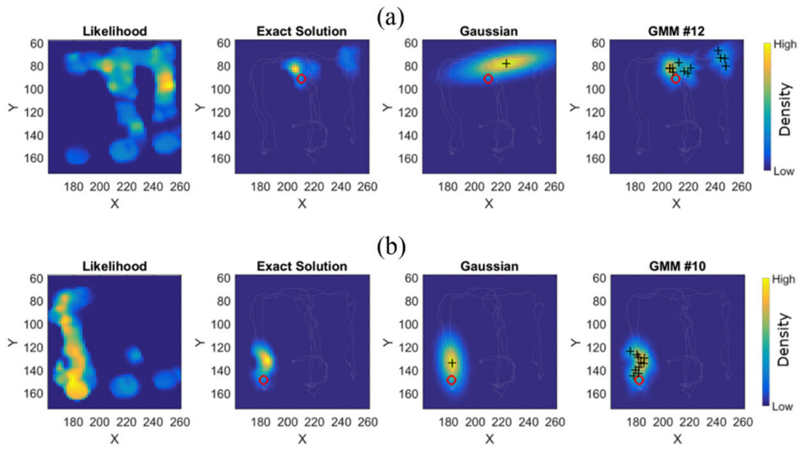

Fig. 1.

2-D decoding result using three different estimation methods at two different time points (a) and (b). The leftmost figures show the instantaneous likelihood at that time point given the observed spiking activity. The second figures from left show the exact solution computed using a 2×2 cm grid. The third figures show the decoding result using the Gaussian approximation, and the rightmost figures show the approximate GMM solution. Each + represents the mean of a mixture component, and numbers in parentheses denote the number of mixtures. The red circle denotes the rat's actual video-tracked position on the maze, and the grey lines represent the rat's entire path throughout the session. In these figures, areas with a higher likelihood are shown in yellow while areas with a lower likelihood are shown in dark blue.

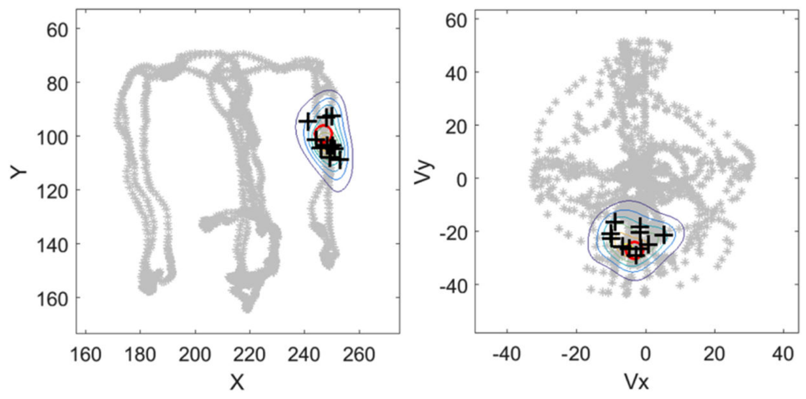

Fig. 2.
4-D decoding result using the approximate GMM filter. The left panel shows the marginal distribution over position, and the right panel shows the marginal distribution over velocity. The red circles denote the rat’s actual position and velocity. The + signs indicate the means of the 11 mixture components for this time point.

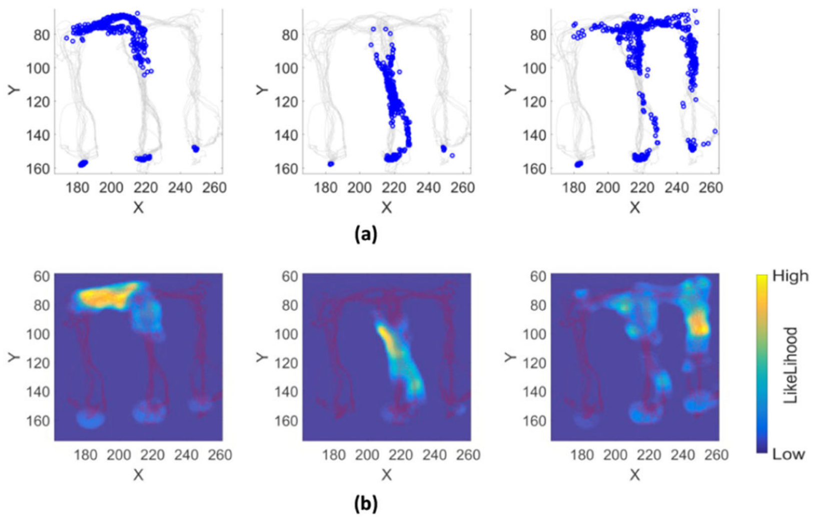

Fig. 3.
Spiking patterns of multiple place cells and their corresponding likelihood functions. (a) Spiking pattern of three different place cells. Each cell fires on a different section of the maze, and its receptive field has a different topology. In the figure, blue marks represent spikes and gray curves are the rat's movement trajectory through the maze. (b) The contribution to the likelihood function over position when each of these place cells fire. The topology of place cells' receptive fields is different, and they are not necessarily unimodal.

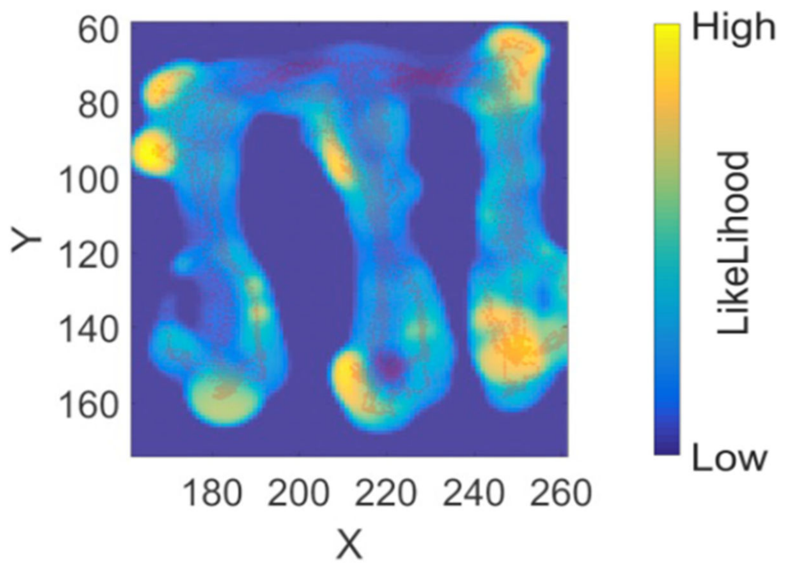

Fig. 4.
Likelihood function on non-spike times. The likelihood expands over the whole maze, and it is non-zero on almost every point on the maze. Note that because of the penalty term, the likelihood everywhere outside of the maze is close to zero.

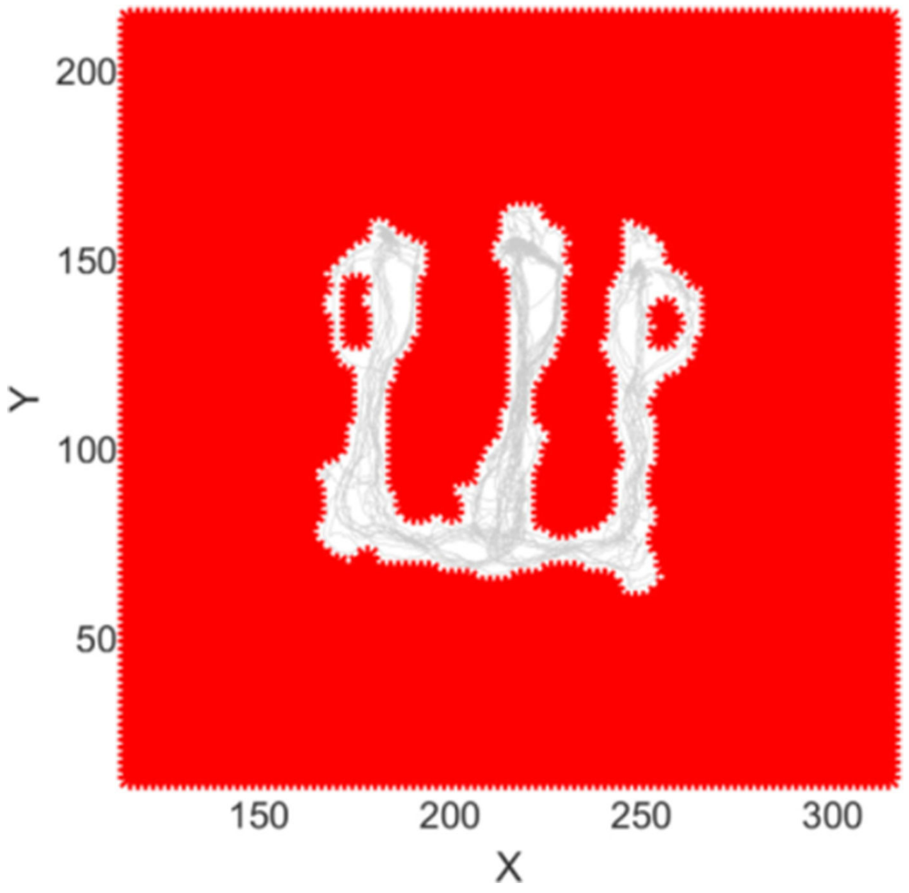

Fig. 5.
The red area shows the penalty area used in the likelihood function to impose a topology constraint. In the penalty area, the penalty term is set to a small number.

TABLE I

Performance Metrics Using Different 2-D and 4-D Decoder Methods

Method	RMSE2D	RMSE4D	95% HPD Coverage	Average Mixtures	95% HPD Normalized	Area Total Runtime (Sec)
Exact [9]	14.03	-	91.04%	-	1	1.48e5
Particle Filter [31]	19.95	-	NA	-	NA	2.01e5
Gaussian [15]	27.3	-	62.39%	i	1.55	1.75e5
Approx. GMM (2D)	17.21	-	85.67%	11.56	1.41	1.99e5
Approx. GMM (4D)	15.66	18.68	87.89%	10.91	1.96	2.48e5

RMSE is a measure of error between the decoded and actual rat position. 95% HPD is the coverage of the 95% highest probability density region of the computed posterior distribution of the rat's actual position. The last row shows performance of the 4D GMM decoder in both 2D and 4D spaces. The gaussian approximation uses our proposed methodology, limited to a single gaussian component at each time point. total runtime provides the whole processing time to run the decode over all 4099 sample data points. Note that the exact solution runs with a coarse resolution of 2 cm over each axis.

TABLE II

Complete Solution Processing Steps for Time Step *k*

```
Initialization:
  set $T$ to 1000. $T$ is dispersion threshold
    1    for s from 1 to $S$ do
             $\boldsymbol{\mu}_{os} = \boldsymbol{A}\,\boldsymbol{\mu}_s$
             $\boldsymbol{\Sigma}_{os} = \boldsymbol{A}\,\boldsymbol{\Sigma}_s\boldsymbol{A}' + \boldsymbol{Q}$
             $\pi_{os} = \pi_s$
           end for
    2    if time step $k$ is a non-spike time then
             call equations (11)–(13) to update $(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s, \boldsymbol{\pi}_s)$ for time $k$
             check eigenvalues of $\boldsymbol{\Sigma}_s$ matrices and find the largest one
           end if
    3    if the largest eigenvalue is larger than $T$ then
             set mode to 1
           else if
             set mode to 0
           end if
    4    if time step $k$ is a spike time or mode is equal to 1 then
           call equation (14)
           call equation (15)
           call the EM algorithm, equations (16)–(23)
           build new $(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s, \boldsymbol{\pi}_s)$ for time $k$
           end if
    5    return new mixture components $(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s, \boldsymbol{\pi}_s)$ for time $k$
```

TABLE III

Performance Result Using Multiple Data Sessions

Data (Rat, Session, # Cell)	Exact (RMSE, HPD%)	Particle Filter (RMSE)	GMM (RMSE, HPD%)	Gaussian Approximation (RMSE, HPD%)	4D GMM (RMSE, HPD%)
(1,1,63)	13.8, 86	14.8	15.1,89	24.9, 67	13.8, 87
(1,2,54)	24.1,74	27.3	26.2, 65	48.0, 16	22.7, 83
(2,1,84)	14.0, 91	20.0	17.2, 86	27.3, 62	15.7, 88
(2,2,84)	25.7, 79	26.8	26.6, 72	27.6,71	26.3, 79

2D and 4D decoding results using different methodologies are presented for 4 different datasets (2 rats, 4 sessions). Sessions 1 and 2 for rat 1 occur on different days and thus we might have a different number of cells. The number of cells identified from the first rat in the second session dropped from 63 to 54.