

# Supplemental Information

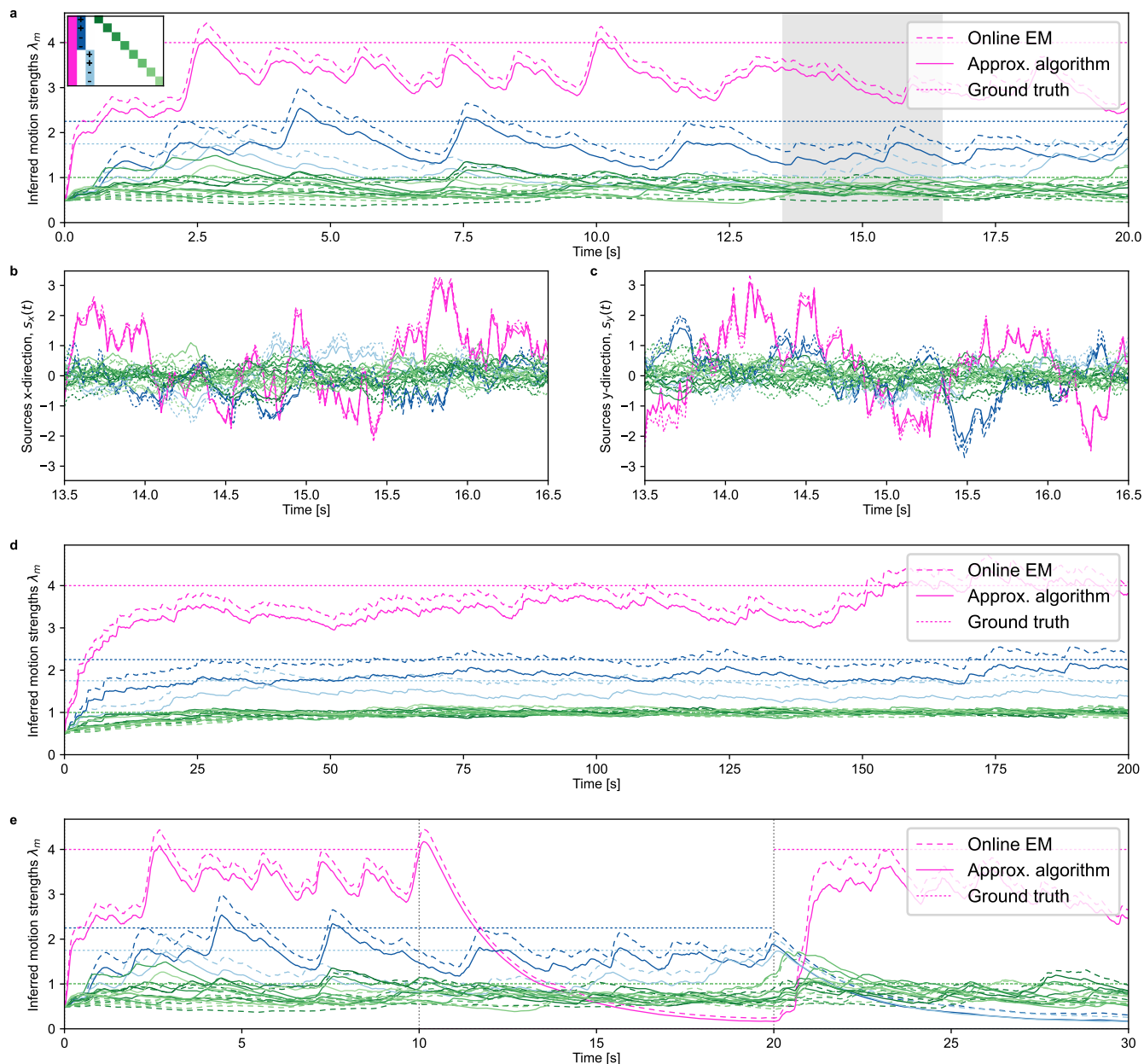
## Structure in motion: visual motion perception as online hierarchical inference

Johannes Bill, Samuel J. Gershman, Jan Drugowitsch

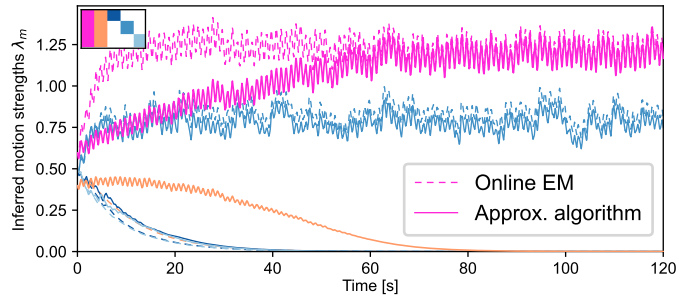
### Contents

<b>1</b>	<b>Supplemental figures</b>	<b>2</b>
<b>2</b>	<b>Generative model of structured motion</b>	<b>8</b>
2.1	Composition of observable velocity from motion motifs . . . . .	8
2.2	Generation of observable velocities from stochastic, latent motion sources . . . . .	8
2.3	Motion structure . . . . .	9
2.4	Extension to multiple spatial dimensions . . . . .	9
2.5	Polar coordinates: rotational and radial motion . . . . .	9
<b>3</b>	<b>Online hierarchical inference algorithm</b>	<b>10</b>
3.1	Inference via the Expectation-Maximization algorithm . . . . .	10
3.1.1	Inference of motion sources for a given structure (E-step) . . . . .	10
3.1.2	Inference of motion strengths and sparsity prior (M-step) . . . . .	12
3.1.3	Extension to multiple spatial dimensions . . . . .	13
3.2	Continuous-time, online inference . . . . .	13
3.2.1	Equivalent formulation of the generative model and inference using natural parameters . . . . .	13
3.2.2	Continuous-time dynamics of natural parameters . . . . .	13
3.2.3	Simultaneous online inference of motion sources and structure . . . . .	14
3.3	Adiabatic approximation for prediction error-based inference . . . . .	15
3.3.1	Prediction error-based formulation . . . . .	15
3.3.2	Convergence approximation on the posterior precision . . . . .	16
3.3.3	Analytic solution for diagonal covariance matrices . . . . .	16
3.3.4	Putting it together: neuro-friendly algorithm for online structure inference . . . . .	17
3.3.5	A pleasant note on inference of rotational and radial motion . . . . .	17
<b>4</b>	<b>Extensions of the online algorithm</b>	<b>17</b>
4.1	Non-isotropic observation noise and missing observations . . . . .	17
4.2	Heterogeneous time constants . . . . .	18
4.3	Interaction priors capturing feature compatibility . . . . .	18
4.4	Detecting motion components that had decayed to baseline . . . . .	19
4.5	Learning the motion components on long time-scales . . . . .	19
<b>5</b>	<b>Neural network implementation</b>	<b>20</b>
5.1	Aims and assumptions . . . . .	20
5.2	Linearly decodable variables . . . . .	20
5.3	Motion structure inference via at-most quadratic operations . . . . .	21
5.4	Neural dynamics for integrating linear and quadratic differential equations . . . . .	21
5.5	Recurrent network model for online motion structure inference . . . . .	22
5.6	Neural coding of the input: an example for area MT . . . . .	24
<b>6</b>	<b>Captions to the supplemental videos</b>	<b>27</b>

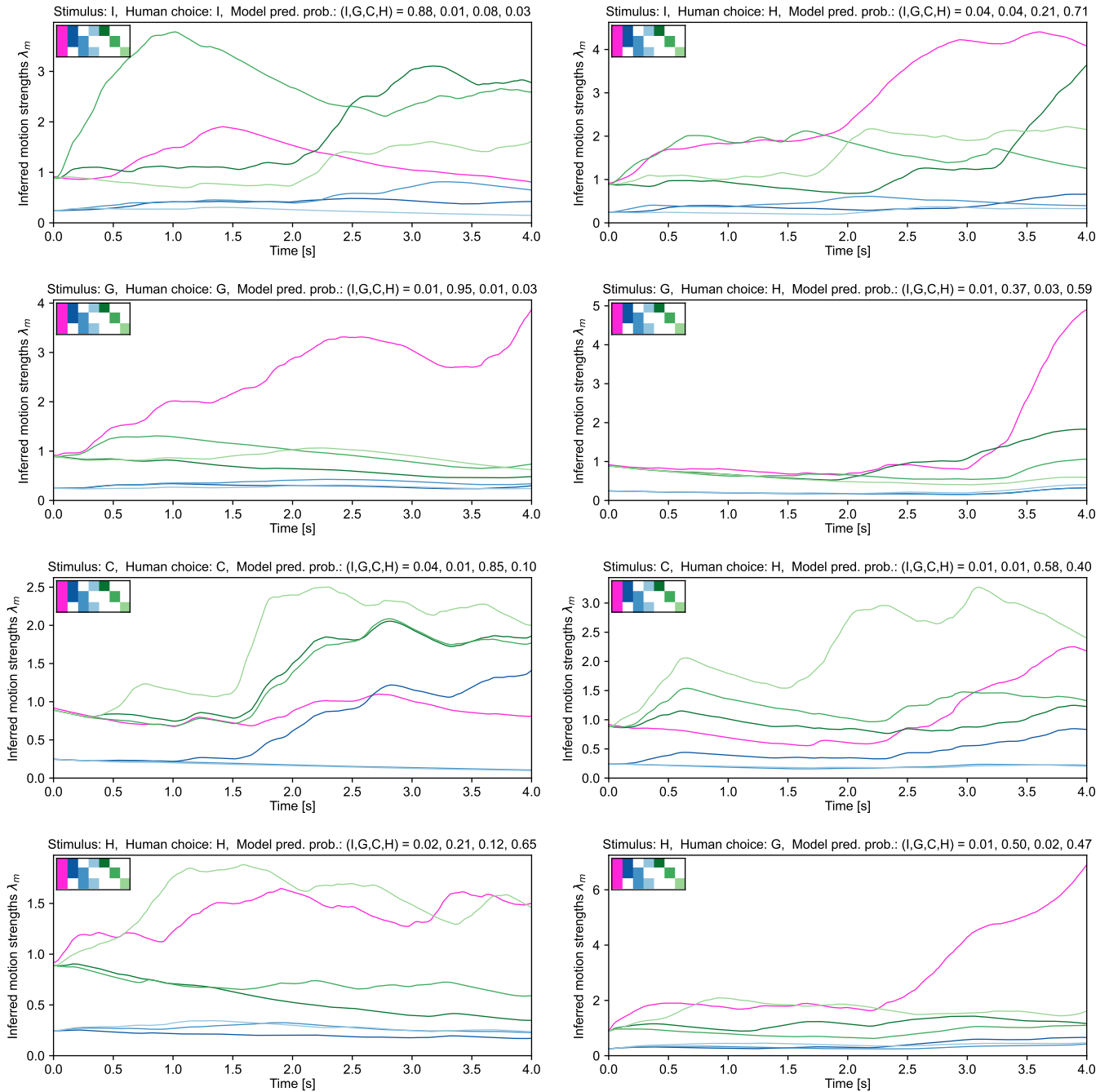
## 1. Supplemental figures



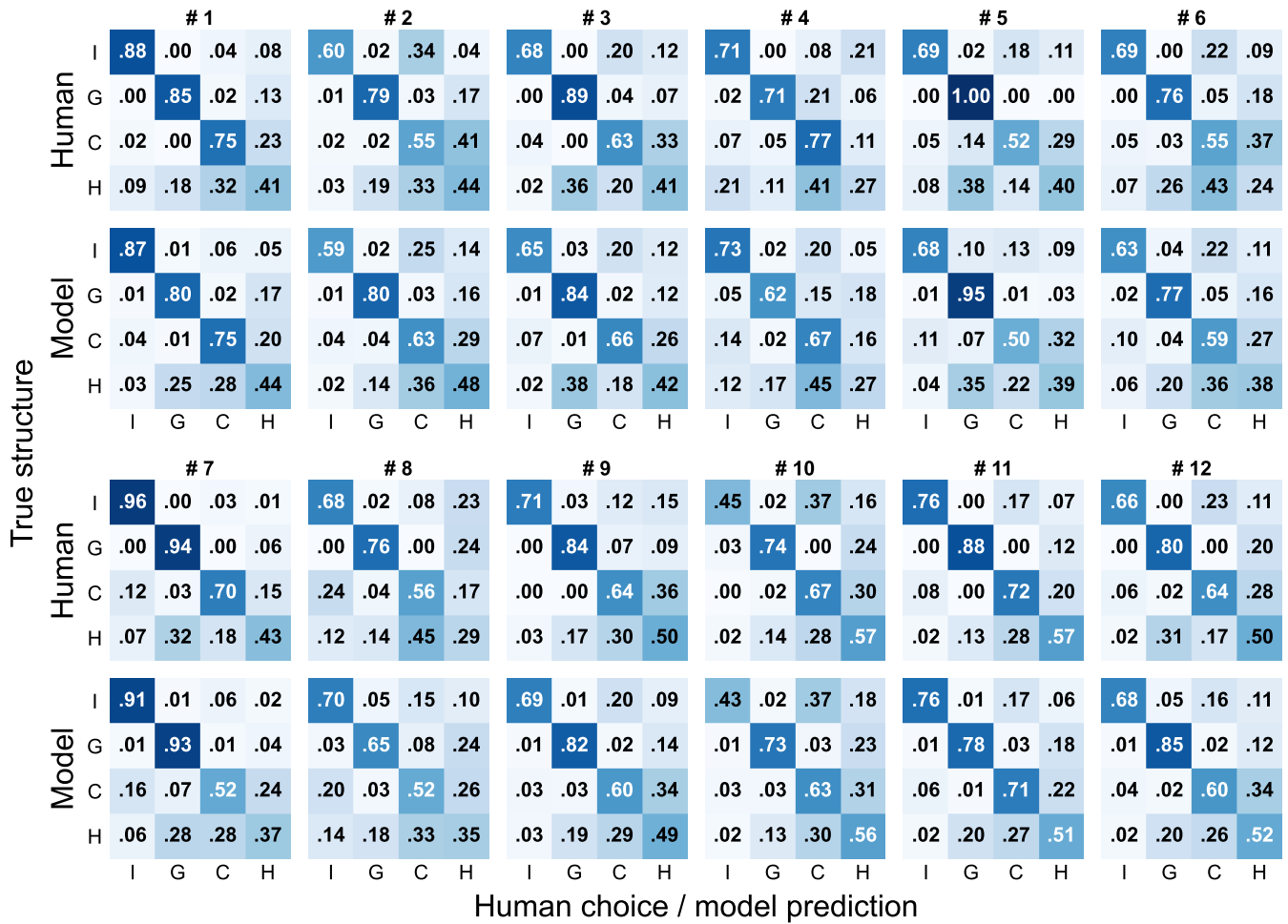
**Figure S1. The online algorithm correctly recovers the structure and motion sources of presented input. (a)** Inferred motion strengths by the online algorithm. Two-dimensional input was generated from the generative model for a deeply nested structure with shared motion (pink;  $\lambda=4$ ), two separate groups of counter-rotating sub-groups (dark- and light-blue;  $+/-$  in the inset indicates  $C_{km} = +1/-1$ ;  $\lambda=2.25$  and  $1.75$ ), and eight individual motions (greens;  $\lambda=1$ ). Other parameters are the default parameters for object-indexed experiments (see table in the *Methods* section of the main paper). Shown are the inferred strengths for the approximate online algorithm (solid lines; given by eqn. (1)–(3) of the main text), the more accurate, but computationally also more complex online EM algorithm (dashed lines; given by eqn. (29), (30), and (35) of the Supplemental Information), and the ground truth (dotted lines). The approximate algorithm yields results similar to the reference online EM algorithm. Both algorithms underestimate the motion strengths due to the sparsity prior  $p(\lambda^2)$ . **(b)** Inferred motion sources, x-direction for the highlighted duration of the simulation in panel a. Same color key as in panel a. **(c)** Same as panel b, but for the y-direction. **(d)** Repetition of the simulation in panel a, but with 10x longer time constant  $\tau_\lambda$ , longer run time, and uniform prior over the motion strengths. The underestimation in the reference algorithm vanishes; the approximate algorithm maintains its approximation quality. **(e)** Repetition of the simulation in panel a, but with a temporally changing structure. After 10 s, the shared component is switched off in the input. After 20 s, the shared component is re-introduced, but the groups are switched off. Both inference algorithms successfully detect these changes.



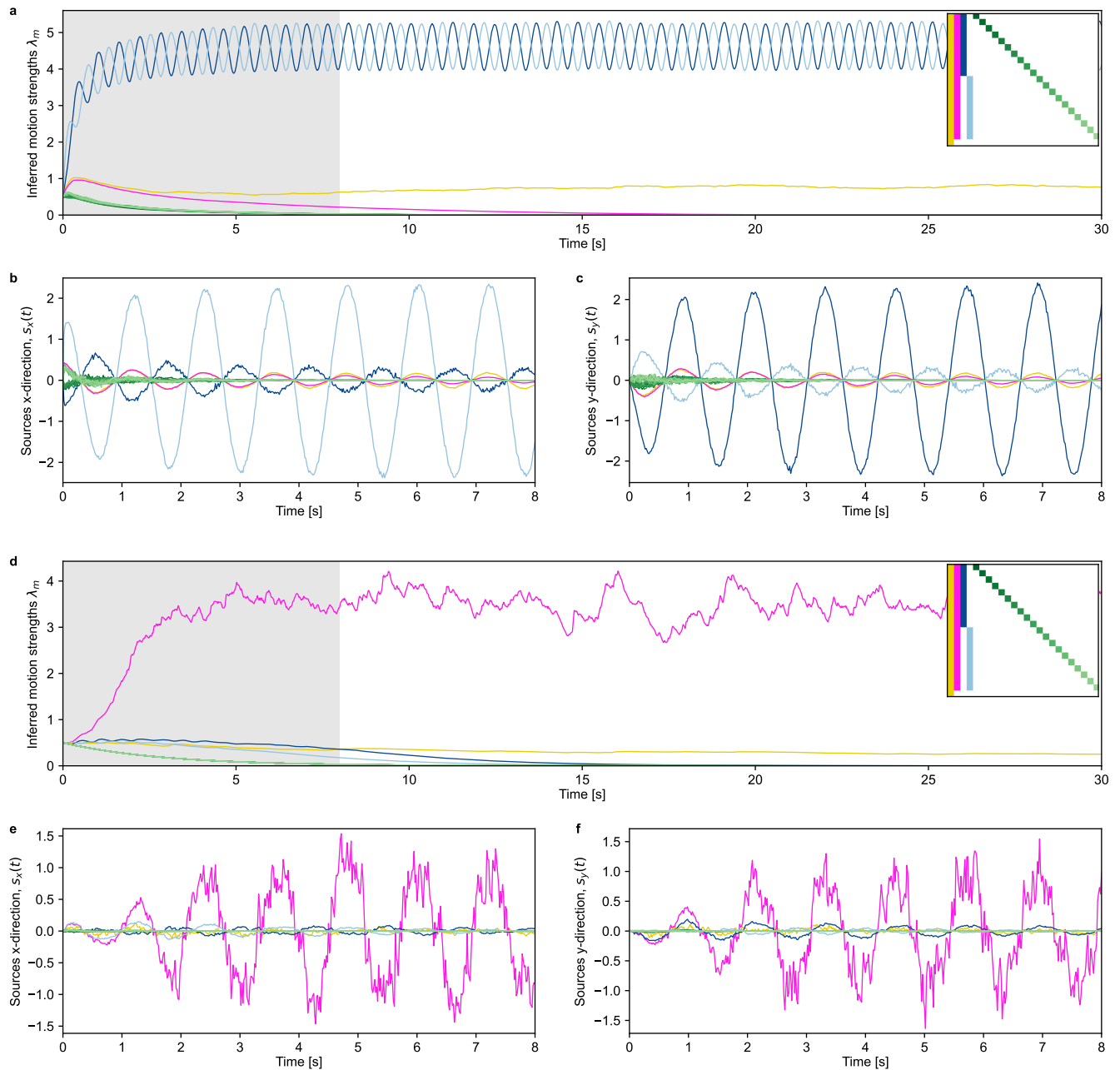
**Figure S2. The online algorithm prefers simple structures, recruiting only necessary components from the reservoir.** Shown is a repetition of the Johansson experiment from Figure 2c, yet with a duplicated shared motion component in the observer model (pink and orange, see inset in the top-left). A small difference at initialization ( $t=0$ ) between the two components widens, such that eventually only one component is recruited and the other one is dismissed. This preference for simpler structures is a direct consequence of the sparsity-inducing Jeffreys prior. If a uniform prior had been used, both shared components would have been maintained (not shown). Furthermore, we notice that the reference online EM algorithm converges more rapidly than the approximate algorithm. The reason is found in the posterior covariance matrix,  $\Sigma$ , which is fully computed for online EM according to eqn. (29) and in which the off-diagonal element between the two shared components introduces competition during the credit assignment in eqn. (39) (uncertainty in the two sources is negatively correlated, leading to a negative matrix element in  $\Sigma$ ).



**Figure S3.** Examples of motion structure inference for trials from (Yang et al., 2021). Shown are traces for  $\lambda(t)$  for eight example trials of participant #1. Axes titles state the ground truth, the participant's classification, and the predicted choice probabilities of the model. *Left column:* Trials of each structure which were correctly classified by the human participant. *Right column:* Trials of each structure which were incorrectly classified by the human participant.



**Figure S4. The online algorithm captures participant-specific error patterns in the data from (Yang et al., 2021).** Shown are the confusion matrices for all 12 participants along with the cross-validated predictions of our model. The model captures participant-specific patterns, such as general performance levels; the preferential misclassification of hierarchical motion (H) as either global (G) or clustered (C); and the asymmetry between the I-C and C-I elements.



**Figure S5. Detailed evolution of all motion strengths and sources in the illusion from Lorenceau.** (a) Motion strengths,  $\lambda(t)$ , in the low-noise condition. Motion is decomposed into a deep hierarchy of self-motion (yellow), shared motion (magenta), group motion (dark and light blue), and individual motion of all  $K=20$  dots (greens). In the generative model, self-motion affects the velocities of all dots and of the zero-mean vestibular input (lowest row in the  $C$ -matrix inset). (b) Inferred x-direction,  $\mu_x(t)$ , of all motion sources in the low-noise condition, for the highlighted duration in panel a. (c) Same as panel b, but for the y-direction. (d–f) Same as panels a–c, but for the high-noise condition.

## List of used variables

Variable	Description	Variable	Description
$v = (v_1, \dots, v_K)^\top$	Observable velocity	$s = (s_1, \dots, s_M)^\top$	Latent motion source
$v_k = v_{k,d}(t)$	Dim. $d$ and time $t$ often suppressed	$s_m = s_{m,d}(t)$	Dim. $d$ and time $t$ often suppressed
$\hat{v} = (\hat{v}_1, \dots, \hat{v}_K)^\top$	Noise-free velocity	$\lambda = (\lambda_1, \dots, \lambda_M)^\top$	Motion strength
$\mu = (\mu_1, \dots, \mu_M)^\top$	Mean vector in $s$ -posterior	$\epsilon = (\epsilon_1, \dots, \epsilon_K)^\top$	Prediction error
$\Sigma = \Omega^{-1}$	Covariance in $s$ -post. and inv. precision	$f_\Sigma$	Adiabatic approx. of $\sigma^2$
$\sigma^2$	Vect. of diag. elements $(\Sigma_{11}, \dots, \Sigma_{MM})^\top$	$C$	Component matrix of shape $(K \times M)$
$D$	no. dimensions	$c_m$	$m$ -th column of $C$
$\text{diag}[x]$	Diag. matrix over some vec. $x$	$\langle f(x) \rangle_{p(x)}$	Expectation of $f(x)$ under $p(x)$
$\tau_s$	Time const. for $s$ -inference (OU process)	$J$	Interaction prior on motion comp.
$\tau_\lambda$	Time const. for $\lambda$ -inference	$\delta t$	Obs. interval (inverse frame rate)
$\tau_\epsilon$	Time const. for pred. err.	$\delta(t), \delta_{ij}$	Dirac and Kronecker delta
$t_{\text{obs}}$	Obs. time point in cont.-time (CT)	$\partial_t$	Partial derivative (here, w.r.t. $t$ )
$t$	No. time steps (DT); <u>Or:</u> time (CT)	$\nu$	No. of $\mu$ -pseudo obs. (hyperprior)
$\mathcal{I}\chi(v, \kappa^2)$	Scaled inverse chi-squared distribution	$\kappa$	Val. of pseudo observations
$\sigma_\rho$	MT width of speed tuning	$\kappa_\alpha$	MT precision of direction tuning
$\sigma_{\text{obs}}$	Observation noise ( $\delta t$ -independent)	$\mathcal{Q}(\lambda)$	EM expect. compl. data log-likelihood
$\tilde{\sigma}_{\text{obs}}$	Disc.-time obs. noise ( $=\sigma_{\text{obs}}/\sqrt{\delta t}$ )	$S$	Structure $I, G, C, H$ in (Yang et al., 2021)
$R$	Radial rec. field loc.	$\beta$	Inv. temp. in (Yang et al., 2021)
$\vartheta$	Angular rec. field location	$b_G, b_C, b_H$	Biases in (Yang et al., 2021)
$s_r, s_\varphi$	Polar sources: radial and angular velocity	$\pi_L$	Lapse prob. in (Yang et al., 2021)
$\rho$	Speed in MT tuning func.	$\alpha$	Direction (angle) in MT tuning
$n_\rho, n_\alpha, N_\rho, N_\alpha$	MT neuron indices, max. values	$\mu_\rho(n_\rho), \mu_\alpha(n_\alpha)$	Tuning center of neuron $(n_\rho, n_\alpha)$
$\psi$	MT max. firing rate multiplier	$I_n(\kappa)$	Modified Bessel function of order $n$
$f, f_\sigma, f_\rho, f_\alpha$	MT tuning func. and sub-functions	$I$	Identity matrix
$W, \bar{W}$	Linear maps in algo. & neural domain	$\mathcal{Q}, \bar{\mathcal{Q}}$	Quad. maps in algo. & neural domain
$b, \bar{b}$	Add. constants in algo. & neural domain	$A, A^\dagger$	Linear decoding matrix & adjoint
$x, y, z$	Variables in generic network derivation	$x_k, y_k$	Spatial locations for 2-dim. stimuli
$r_{\text{inp}}, r_{\text{dis}}, r_{1\text{-to-1}}$	Firing rates in the network model	$\gamma$	Direction repulsion opening angle
$v_{\text{vst}}$	Vestibular self-motion input	$\dot{\varphi}$	Angular (rotational) velocity
$\sigma_{\text{vst}}$	Obs. noise for vestibular input	$\dot{r}$	Radial velocity
$\eta_C$	Learning rate for $C$ -learning		

**Table 1. List of used variables.** In a few cases, we chose to accept notation clashes when confusion is ruled out and the dual use actually facilitates clarity. These cases are indicated with “Or:” in the description.

## Theory of the inference algorithms and the neural network model

In **Section 2**, we introduce the generative model for structured motion. In **Section 3**, we derive the online hierarchical inference algorithm. In **Section 4**, we present optional extensions to the algorithm. In **Section 5**, we derive the recurrent neural network model.

### 2. Generative model of structured motion

The following model of hierarchically structured motion is an adaptation of the generative model from (Bill et al., 2020). We consider  $K$  observable velocities,  $v_{k,d}(t)$ , in  $D$  spatial dimensions. To prevent clutter, we will develop most of the theory for the one-dimensional case,  $D=1$ , and use the vector notation,  $v = (v_1, \dots, v_K)^\top$ . The extension to  $D > 1$  is straightforward and will be covered in a dedicated subsection. We will often suppress the explicit time dependence when confusion is ruled out. In other cases, we may write time as an index,  $v_t$ , when a compact notion is desirable. Observable velocities,  $v$ , are noisy instantiations of noise-free velocities,  $\hat{v}$ , which are generated by  $M$  latent motion sources,  $s_{m,d}(t)$ , as will be specified below. Similar to velocities, we abbreviate  $s = (s_1, \dots, s_M)^\top$ .

#### 2.1 Composition of observable velocity from motion motifs

For the purpose of this work, we restrict the influence of motion source  $s_m(t)$  on velocity  $\hat{v}_k(t)$  to three particularly basic relations: the motion of  $s_m$  can affect  $\hat{v}_k$  either positively (e.g., the latent flock motion on an observed bird's velocity), negatively (e.g., the effect of self-motion on the observed scene), or not at all (e.g., the flock motion's effect on a tree). Formally, we describe these influences in a  $K \times M$  component matrix  $C$ , with  $C_{km} = +1, -1$ , and  $0$  for positive, negative and absent influence, respectively. We will sometimes refer to the columns of  $C$  as motion motif or motion component  $c_m$ . Typically, we have  $M > K$  because every observable can have its individual motion component which exclusively affects this observable.

The overall velocity  $\hat{v}_k$  is the sum of all the motion sources' contributions:

$$\hat{v} = C s . \quad (1)$$

It is easy to see that all tree-like hierarchies can be cast into this form. As a word of warning, the opposite is not true: not every possible  $C$  represents hierarchically structured motion.

Finally, it is worth mentioning that motion components are not necessarily limited to the simple  $C_{km} = \pm 1$  values. For modeling gradual relationships, a motion source could affect some observables stronger than others. For instance, the current of a river could influence the water's velocity in the middle of the river stronger than close to the bank. All subsequent derivations likewise hold for such gradual relationships, as long as they influence the observed velocities linearly.

#### 2.2 Generation of observable velocities from stochastic, latent motion sources

For the generative model, we assume that the motion sources,  $s_m(t)$ , evolve independently according to an Ornstein-Uhlenbeck process, see, e.g., (Gardiner, 2009),

$$ds_m = -\frac{1}{\tau_s} s_m dt + \lambda_m dW_m , \quad (2)$$

where Wiener process  $W_m(t)$  drives changes in  $s_m$  via random forces, motion strength  $\lambda_m \geq 0$  controls the magnitude of these forces, and  $\tau_s > 0$  is the typical time constant of significant changes in  $s_m$ . The equilibrium distribution of this process is a normal distribution with zero mean and variance  $\frac{\tau_s}{2} \lambda_m^2$ :

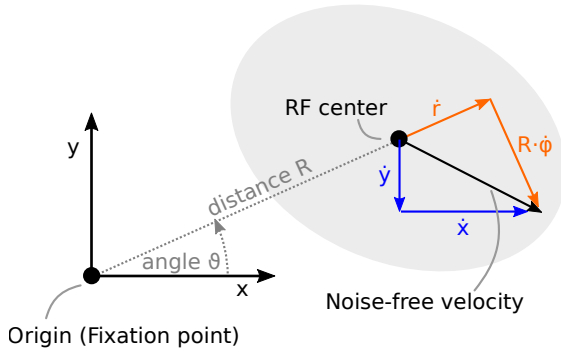
$$\lim_{t \rightarrow \infty} p(s_m(t)) = \mathcal{N}(s_m; 0, \frac{\tau_s}{2} \lambda_m^2) . \quad (3)$$

The Ornstein-Uhlenbeck process is a suitable first-order approximation for modeling real-world motion as it (i) generates continuous trajectories  $s_m(t)$ , (ii) includes a notion of inertia/momentum via its temporal filtering with  $\tau_s$ , (iii) offers an intuitive parameterization by scaling typical velocities linearly in  $\lambda_m$ , and (iv) implements a slow-velocity prior via a normal distribution like in (Weiss, Simoncelli, and Adelson, 2002).

Observable velocities,  $v(t)$ , are generated by composing the latent motion sources to noise-free velocities according to eqn. (1) and then applying independent Gaussian noise:

$$v(t) \sim \mathcal{N}(C s(t), \tilde{\sigma}_{\text{obs}}^2 \mathbf{I}) . \quad (4)$$





$$\begin{aligned}\dot{x} &= \dot{r} \cos \vartheta - \dot{\varphi} R \sin \vartheta \\ \dot{y} &= \dot{r} \sin \vartheta + \dot{\varphi} R \cos \vartheta \\ \dot{r} &= \dot{x} \cos \vartheta + \dot{y} \sin \vartheta \\ \dot{\varphi} &= \frac{1}{R} (-\dot{x} \sin \vartheta + \dot{y} \cos \vartheta)\end{aligned}$$

**Figure S6. Coordinate transformation from polar motion sources to Cartesian velocities for supporting rotational and radial motion motifs.** When receptive field (RF) centers are fixed, rotational motion,  $\dot{\varphi}$ , and radial motion,  $\dot{r}$ , feature fixed linear relations to the resulting Cartesian velocities in x- and y-direction. The angles  $\vartheta$  and  $\varphi$  are measured relative to the x-axis in counter-clockwise direction by convention. Cartesian components are measured in rightward (x) and upward (y) direction.

In foresight of the continuous-time formulation, we denote the observation noise by  $\tilde{\sigma}_{\text{obs}}$  (with a tilde). It will later be adjusted to the frame rate of incoming observations such that the information per unit time remains constant. To keep the derivation tractable, we will ignore the reported velocity-dependence of observation noise in human perception (Weber's law) and treat  $\tilde{\sigma}_{\text{obs}}$  as a constant.

Marginalizing over the stationary distribution of the latent motion sources, eqn. (3), the observable velocities are jointly normally distributed due to their linear dependence on a Gaussian origin, with zero mean and covariance matrix  $\frac{\tau_s}{2} C \text{diag}[\lambda^2] C^T + \tilde{\sigma}_{\text{obs}}^2 I$ , with  $\text{diag}[\lambda^2]$  denoting the diagonal matrix generated from vector  $\lambda^2 = (\lambda_1^2, \dots, \lambda_M^2)^T$ .

### 2.3 Motion structure

The motion strengths  $\lambda$  play a particularly important role in the generative process described by eqn. (2)+(4). For  $\lambda_m = 0$ , dependent motion sources decay to zero, i.e.,  $s_m \rightarrow 0$ . Hence, motion strengths describe the presence ( $\lambda_m > 0$ ) or absence ( $\lambda_m = 0$ ) of motion components, as well as their typical magnitude ( $\langle |s_m| \rangle \propto \lambda_m$ ). In other words, given a reservoir of components,  $C$ , which have been learned to occur in visual scenes in general, the vector  $\lambda$  will describe the structural composition of motion relations in a specific visual scene. Knowing  $\lambda$  therefore is equivalent to knowing the motion structure of the scene.

### 2.4 Extension to multiple spatial dimensions

For  $D > 1$ , we will usually assume that all  $s_{m,d}$  with the same index  $m$  share motion strength  $\lambda_m$  and motion component  $c_m$ , yet each  $s_{m,d}$  follows its own stochastic evolution, i.e., has its own Wiener process,  $W_{m,d}$ , in eqn. (2). This choice reflects that space is isotropic: due to the Gaussianity of each  $s_{m,d}$ , the joint distribution  $p(s_{m,1}, \dots, s_{m,D})$  will be a multivariate Gaussian with covariance matrix  $\frac{\tau_s}{2} \lambda_m^2 I$ , with  $I$  denoting the identity matrix, and is, thus, invariant to rotations in the experimenter's choice of the coordinate system. Again, this is a useful first-order approximation, even though human motion perception has been reported to be not perfectly isotropic (Manning, Thomas, and Braddick, 2018; Moscatelli et al., 2019).

In summary, the generative model of structured motion is characterized by the following set of parameters: number of spatial dimensions  $D$ , motion strengths  $\lambda$ , component matrix  $C$ , time constant  $\tau_s$ , and observation noise  $\tilde{\sigma}_{\text{obs}}$ . Some extensions of this model covering heterogeneous time constants, non-isotropic observation noise and missing observations will be discussed in dedicated subsections in the context of online inference, below.

### 2.5 Polar coordinates: rotational and radial motion

The discussion so far assumed that motion sources affect velocities in a translational (Cartesian) manner, that is, by adding the same vector to all dependent velocities. An important exception are rotational and radial motion in two dimensions ( $D=2$ ) which typically occur in flow field parsing (when moving forward, everything expands on the retina; when tilting your head, everything rotates in the opposite rotational direction).

It turns out that rotational and radial motion sources can be incorporated into our framework of linear velocity generation as per eqn. (1), when the spatial locations of all dependent velocities are fixed. This is, for example, fulfilled in experimental setups using drifting gratings: the drifting grating has motion energy, but its location stays fixed

at the same spot. Note how this setup is different from the perspective commonly taken in physics where velocity entails changes in location. The perspective we take here is geared towards studying brain computation where a group of neurons often processes motion in a certain, fixed area of the inputs. Specifically, this is satisfied in many visual areas including V1, MT, and MST, where neurons feature a fixed spatial receptive field in retinal coordinates. The index  $k$  in  $v_k$  then refers to a fixed location on the retina. We therefore call this condition “location-indexed”.

To derive the generative model for rotational and radial motion in a location-indexed experiment, consider the spatial receptive fields illustrated in **Fig. S6**. Each receptive field (RF) has a fixed center relative to the fovea and is characterized by radial distance  $R$  and angle  $\vartheta$ , which is measured relative to the x-axis in counter-clockwise direction by convention. The generative model can use latent motion sources in radial direction, denoted by  $\dot{r}$  in the figure, and in angular direction, denoted by  $\dot{\varphi}$ . These polar motion sources generate velocities  $\dot{x}$  and  $\dot{y}$  in Cartesian space. Importantly, the (generative) transformation from polar to Cartesian coordinates is linear for each receptive field:

$$\begin{aligned}\dot{x} &= \dot{r} \cos \vartheta - \dot{\varphi} R \sin \vartheta \\ \dot{y} &= \dot{r} \sin \vartheta + \dot{\varphi} R \cos \vartheta .\end{aligned}\tag{5}$$

Consequently, the generative process can be included into the component matrix  $\mathbf{C}$  when each polar motion source affects dependent observables,  $\hat{v}_k$ , in both spatial dimensions. In contrast to the previously discussed translational motion sources, rotational and radial motion sources maintain separate motion components  $c_m$  and motion strengths  $\lambda_m$ :

$$\begin{aligned}c_{k,m=\text{rad},d=x} &= \cos \vartheta_k , & c_{k,m=\text{rot},d=x} &= -R_k \sin \vartheta_k \\ c_{k,m=\text{rad},d=y} &= \sin \vartheta_k , & c_{k,m=\text{rot},d=y} &= +R_k \cos \vartheta_k .\end{aligned}\tag{6}$$

As before, the model adds up the velocities from different sources and applies the observation noise only to the final velocity (in Cartesian space). The example of rotational and radial motion demonstrates how spatial dimensions can be mixed, as long as the coordinate transformation is linear for every observable  $\hat{v}_k$ .

### 3. Online hierarchical inference algorithm

In **Section 3.1**, we develop the hierarchical inference algorithm in a discrete-time, batch formulation, using the Expectation-Maximization (EM) algorithm. In **Section 3.2**, we draw the continuous-time limit to obtain an online algorithm. Finally, in **Section 3.3**, we introduce an adiabatic approximation which reduces the required computations to neuron-friendly operations. This is the online algorithm underlying all simulations of the main paper.

#### 3.1 Inference via the Expectation-Maximization algorithm

Our goal for motion structure inference is to simultaneously infer the value of motion sources  $s(t)$  and the underlying structure  $\lambda$  from a stream of observations  $v_{1:t} = (v_1, \dots, v_t)$  with observations arriving at time steps of duration  $\delta t$  (inverse frame rate). The number of spatial dimensions  $D$ , components  $C$ , time constant  $\tau_s$ , and observation noise  $\tilde{\sigma}_{\text{obs}}$  are assumed to be known (although  $C$  could be learned on longer time scales, see **Section 4.5**). The challenge in this hierarchical inference task is that  $s_t$  and  $\lambda$  are mutually dependent on another:  $\lambda$  acts as a parameter in  $p(s_t | \lambda)$  per eqn. (2), and will therefore affect the posterior  $p(s_t | v_{1:t})$ . On the other hand, inferring the presence of, say, flocking birds ( $\lambda_{\text{flock}} > 0$ ) depends on “perceiving” an instantaneous flock motion,  $s_{\text{flock}}(t)$ , in the first place.

The EM algorithm (Dempster, Laird, and Rubin, 1977) offers a solution to this chicken-and-egg problem. For its application, we leverage the fact that motion sources and strengths change on different time scales,  $\tau_s$  for  $s_t$  and  $\tau_\lambda$  for  $\lambda$ . For  $\tau_\lambda \gg \tau_s$ , we can treat  $\lambda$  as a constant while inferring  $s(t)$ —known as the E-step in EM—and then, in alternation, optimize  $\lambda$  based on the inferred motion strengths—the M-step in EM. In **Section 3.1.1** and **Section 3.1.2**, we will address the E-step and the M-step separately. As before, we will develop the theory in one spatial dimension for notational clarity and then generalize to  $D > 1$ . Further, we will present the derivation in discrete time in a batch formulation. The continuous-time limit will be drawn in **Section 3.2**.

##### 3.1.1 Inference of motion sources for a given structure (E-step)

For the E-step, we aim to infer  $p(s_{1:t} | v_{1:t}; \lambda)$  for a given structure  $\lambda$ , and then compute the expected value of the log-likelihood of the augmented data distribution  $p(v_{1:t}, s_{1:t}; \lambda)$ , see, e.g., Section 9.3. in (Bishop, 2006). For the remainder of this subsection, we will often suppress the explicit dependence on  $\lambda$  to avoid notational clutter. Since

we are interested in an online algorithm, we will use the filtering solution which is obtained from iterative application of temporal propagation to the next observation time,

$$p(\mathbf{s}_{t-1} | \mathbf{v}_{1:t-1}) \longrightarrow p(\mathbf{s}_t | \mathbf{v}_{1:t-1}) , \quad (7)$$

and integration of the next observation,

$$p(\mathbf{s}_t | \mathbf{v}_{1:t}) \propto p(\mathbf{s}_t | \mathbf{v}_{1:t-1}) p(\mathbf{v}_t | \mathbf{s}_t) . \quad (8)$$

Propagation, eqn. (7), is performed by propagating the density according to the stochastic process in eqn. (2). Mathematically, this is done by convolving  $p(\mathbf{s}_{t-1} | \mathbf{v}_{1:t-1})$  with the Gaussian transition density  $p(\mathbf{s}_t | \mathbf{s}_{t-1})$  of the Ornstein-Uhlenbeck process. Integration, eqn. (8), is the application of Bayes rule using the emission model of eqn. (4).

For linear stochastic dynamics with a Gaussian emission model, i.e., the present case, the posterior will always be a multivariate Gaussian with some mean  $\boldsymbol{\mu}_t$  and covariance  $\boldsymbol{\Sigma}_t$ :  $p(\mathbf{s}_t | \mathbf{v}_{1:t}) = \mathcal{N}(\mathbf{s}_t; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ . Kalman filtering (Kalman, 1960) is one possible algorithm for calculating the posterior moments  $\boldsymbol{\mu}_t$  and  $\boldsymbol{\Sigma}_t$ , and we refer the interested reader to the Supporting Information of (Bill et al., 2020) for explicit forms of the Kalman filter's state transition matrix and process noise covariance matrix. For the present work, we will employ a more elegant, continuous-time solution that is equivalent to the Kalman-Bucy filter (Kalman and Bucy, 1961; Jazwinski, 2007) for calculating  $\boldsymbol{\mu}_t$  and  $\boldsymbol{\Sigma}_t$ . The derivation, which is provided in **Section 3.2**, will furthermore facilitate a neuro-friendly approximate implementation (**Section 3.3**). For now, let us assume that the posterior moments,  $\boldsymbol{\mu}_{1:t}$  and  $\boldsymbol{\Sigma}_{1:t}$ , have been computed by whichever method of choice.

**E-step.** We compute the expected value of the log-likelihood of the augmented data distribution,<sup>1</sup>

$$Q(\boldsymbol{\lambda}) = \langle \log p(\mathbf{v}_{1:t}, \mathbf{s}_{1:t}; \boldsymbol{\lambda}) \rangle_{p(\mathbf{s}_{1:t} | \mathbf{v}_{1:t})} \quad (9)$$

$$= \sum_{j=1}^t \langle \log p(\mathbf{v}_j | \mathbf{s}_j; \boldsymbol{\lambda}) + \log p(\mathbf{s}_j | \mathbf{s}_{j-1}; \boldsymbol{\lambda}) \rangle_{p(\mathbf{s}_j, \mathbf{s}_{j-1} | \mathbf{v}_{1:t})} \quad (10)$$

$$\approx \text{ind.} \sum_{j=1}^t \langle \log p(\mathbf{v}_j | \mathbf{s}_j; \boldsymbol{\lambda}) + \log p(\mathbf{s}_j; \boldsymbol{\lambda}) \rangle_{p(\mathbf{s}_j | \mathbf{v}_{1:j})} \stackrel{\text{def.}}{=} \sum_{j=1}^t Q_j(\boldsymbol{\lambda}) , \quad (11)$$

where, at the third equality, we have made two approximations, namely, that (i) consecutive observations were independent, and (ii) we take the expectation w.r.t. the filtering posterior,  $p(\mathbf{s}_j | \mathbf{v}_{1:j})$ , rather than the smoothing density,  $p(\mathbf{s}_j | \mathbf{v}_{1:t})$ . The first approximation will be corrected for during the M-step in **Section 3.1.2**, where we will weigh the likelihood term as if it was comprised of only  $t \cdot \delta t / \tau_s$  independent samples when combining it with a sparsity prior on  $\boldsymbol{\lambda}$ . This correction is justified because it is equivalent to considering only a sparse subsample of observations that each lie  $\tau_s$  apart. Consecutive observations in this subsample are almost decoupled since the Ornstein-Uhlenbeck process in eqn. (2) decorrelates the motions sources at time scale  $\tau_s$ . The second approximation—filtering rather than smoothing—is a modeling assumption for what information is employed by an online agent, e.g., a human observer.

The  $Q_j(\boldsymbol{\lambda})$  can be calculated analytically:

$$Q_j(\boldsymbol{\lambda}) = \langle \log p(\mathbf{v}_j | \mathbf{s}_j; \boldsymbol{\lambda}) + \log p(\mathbf{s}_j; \boldsymbol{\lambda}) \rangle_{p(\mathbf{s}_j | \mathbf{v}_{1:j})} \quad \leftarrow \text{with } p(\mathbf{s}_j | \mathbf{v}_{1:j}) = \mathcal{N}(\mathbf{s}_j; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \quad (12)$$

$$= \langle -\frac{1}{2}(\mathbf{v}_j - \mathbf{C}\mathbf{s}_j)^\top \tilde{\sigma}_{\text{obs}}^{-2} \mathbf{I} (\mathbf{v}_j - \mathbf{C}\mathbf{s}_j) - \frac{1}{2} \mathbf{s}_j^\top \text{diag}[\frac{\tau_s}{2} \boldsymbol{\lambda}^2]^{-1} \mathbf{s}_j - \frac{1}{2} \log |\tilde{\sigma}_{\text{obs}}^2 \mathbf{I}| - \frac{1}{2} \log |\text{diag}[\frac{\tau_s}{2} \boldsymbol{\lambda}^2]| \rangle_{p(\mathbf{s}_j | \mathbf{v}_{1:j})} + \text{const.} \quad (13)$$

$$= \sum_{m=1}^M -\frac{1}{\tau_s} \frac{\mu_{j,m}^2 + \sigma_{j,m}^2}{\lambda_m^2} - \frac{1}{2} \log \lambda_m^2 \quad [\text{dropping } \boldsymbol{\lambda}\text{-independent terms and using } \langle s^2 \rangle_{\mathcal{N}(s; \mu, \sigma^2)} = \mu^2 + \sigma^2] \quad (14)$$

with  $\sigma_{j,m}^2 := \Sigma_{j,mm}$ ,  $|\cdot|$  denoting the determinant, and the  $\boldsymbol{\lambda}$ -independent terms have been dropped because they will not play a role in the maximization w.r.t.  $\boldsymbol{\lambda}$  in **Section 3.1.2**. For the full  $Q(\boldsymbol{\lambda})$ , we thus obtain:

$$Q(\boldsymbol{\lambda}) = \sum_{j=1}^t Q_j(\boldsymbol{\lambda}) = -\frac{t}{\tau_s} \sum_{m=1}^M \frac{\langle \mu_{j,m}^2 + \sigma_{j,m}^2 \rangle_j}{\lambda_m^2} + \frac{\tau_s}{2} \log \lambda_m^2 , \quad (15)$$

<sup>1</sup>In the 2nd line, we implicitly assume, for mathematical rigor only, that there is an initial distribution,  $p(\mathbf{s}_{t=0})$ , which is absorbed again in the 3rd line.

with  $\langle \cdot \rangle_j$  being the time average over the batch.

### 3.1.2 Inference of motion strengths and sparsity prior (M-step)

For the M-step,  $Q(\lambda)$  is maximized w.r.t.  $\lambda$  to obtain the maximum likelihood (ML) solution. Here, we will make use of the additional freedom to impose a prior distribution,  $p(\lambda)$ , on the motion strengths, see, e.g., Section 9.4. in (Bishop, 2006). First, we will introduce a family of prior distributions which reflect our knowledge that from a reservoir of motion components most components will be absent or small in any given scene (sparsity prior). Then, we will perform the M-step to derive the maximum a posteriori (MAP) solution.

**Sparsity prior.** In foresight of the M-step, we formulate the prior over  $\lambda^2$  (instead of  $\lambda$ ) and choose

$$p(\lambda^2; \nu, \kappa^2) = \prod_{m=1}^M \mathcal{I}\chi(\lambda_m^2; \nu_m, \kappa_m^2) \quad \text{with} \quad \mathcal{I}\chi(\lambda^2; \nu, \kappa^2) = \frac{1}{\lambda^{(2+\nu)}} \exp \left[ -\frac{\nu\kappa^2}{2\lambda^2} - A(\nu, \kappa^2) \right]. \quad (16)$$

$\mathcal{I}\chi(\lambda^2; \nu, \kappa^2)$  denotes the density of the scaled inverse chi-squared distribution which is the conjugate prior to a normal distribution with known mean and unknown variance. Conceptually, this is exactly our task at hand: we know that  $\langle s_m \rangle = 0$  in the generative model, but we have to estimate its variance,  $\langle s_m^2 \rangle$ , which is controlled by  $\lambda_m^2$ . As will become obvious in the M-step, the (hyper-)parameters,  $\nu$  and  $\kappa^2$ , will take the role of pseudo-counts and pseudo-observations, respectively. The log-partition,  $A(\nu, \kappa^2) = \log \Gamma(\frac{\nu}{2}) - \frac{\nu}{2} \log \frac{\nu\kappa^2}{2}$ , will have no effect on the MAP estimate.

Two choices of  $\nu$  and  $\kappa^2$  are of particular interest. For  $\nu = \kappa^2 = 0$ , we have  $p(\lambda^2) \propto 1/\lambda^2$  which is a non-informative (Jeffreys) prior on the variance of  $s^2$ . For  $\nu = -2, \kappa^2 = 0$ , we have  $p(\lambda^2) \propto 1$  which is a uniform prior. The latter choice of hyper-parameters will turn the MAP estimate into the ML estimate.

**M-step.** To maximize  $Q$ , we find the roots of its  $\lambda_m^2$ -derivatives:

$$0 \stackrel{!}{=} \frac{d}{d\lambda_m^2} (Q(\lambda) + \log p(\lambda^2; \nu, \kappa^2)) = \frac{t}{\tau_s} \left( \frac{\langle \mu_{j,m}^2 + \sigma_{j,m}^2 \rangle_j}{(\lambda_m^2)^2} - \frac{\tau_s}{2} \frac{1}{\lambda_m^2} \right) + \frac{\nu_m \kappa_m^2}{2(\lambda_m^2)^2} - \frac{1 + \frac{\nu_m}{2}}{\lambda_m^2} \quad (17)$$

$$\Rightarrow t \langle \mu_{j,m}^2 + \sigma_{j,m}^2 \rangle_j + \frac{\tau_s}{2} \nu_m \kappa_m^2 = t \frac{\tau_s}{2} \lambda_m^2 + \frac{\tau_s}{2} (2 + \nu_m) \lambda_m^2 \quad (18)$$

$$\Rightarrow \frac{\tau_s}{2} \lambda_m^2 = \frac{t \langle \mu_{j,m}^2 + \sigma_{j,m}^2 \rangle_j + \frac{\tau_s}{2} \nu_m \kappa_m^2}{2 + \nu_m + t}. \quad (19)$$

Eqn. (19) highlights several intuitive properties of motion structure inference. First, the MAP value of motion strength  $\lambda_m^2$  only depends on the inferred posterior moments of the corresponding motion source  $s_m$ , that is, there is no cross-talk between motion sources  $s_m$  and  $s_{m'}$ . Second, recalling that  $\frac{\tau_s}{2} \lambda_m^2$  is the expected long-term variance of  $s_m$  according to eqn. (3), eqn. (19) tells us to match this expected variance,  $\frac{\tau_s}{2} \lambda_m^2$ , to the observed variance,  $\langle \mu_{j,m}^2 + \sigma_{j,m}^2 \rangle_j$ , of the inferred motion source *over time*.<sup>2</sup> Third, hyper-parameter  $\kappa_m$  plays the role of an average pseudo-observation,  $\frac{\tau_s}{2} \kappa_m^2 = \langle \mu_m^2 + \sigma_m^2 \rangle$ , which is then weighted as  $\nu_m$  pseudo-samples against the  $t$  observed data samples. Thus,  $\kappa_m$  describes (a priori) typical values of  $\lambda_m$ . (The summand 2 in the denominator of eqn. (19) is a relict of the scale-invariant Jeffreys prior.) Finally, a uniform hyper-prior,  $\nu_m = -2, \kappa_m = 0$ , yields straightforward variance matching as the ML solution.

We conclude the M-step, by correcting eqn. (19) for the fact that the  $t$  data samples actually only represent  $t \cdot \delta t / \tau_s$  independent samples, as promised in **Section 3.1.1**,

$$\frac{\tau_s}{2} \lambda_m^2 = \frac{\frac{t \cdot \delta t}{\tau_s} \langle \mu_{j,m}^2 + \sigma_{j,m}^2 \rangle_j + \frac{\tau_s}{2} \nu_m \kappa_m^2}{2 + \nu_m + \frac{t \cdot \delta t}{\tau_s}}. \quad (20)$$

Eqn. (20) is the batch solution to motion structure inference which we will build on for the continuous-time formulation in **Section 3.2**.

<sup>2</sup>This is to be distinguished from the posterior's instantaneous uncertainty  $\sigma_{i,m}^2$ .

### 3.1.3 Extension to multiple spatial dimensions

When  $D > 1$ , we typically assume that each  $\lambda_m$  controls the variance of the  $s_{m,d}$  in all spatial dimensions  $d$  (see [Section 2.4](#)). When going through the derivation of the EM algorithm, the decisive changes happen in eqn. (15): (i) the expectation now runs over all spatial dimensions, i.e.,  $\langle \sum_{d=1}^D \mu_{j,m,d}^2 + \sigma_{j,m,d}^2 \rangle_j$ ; (ii) the log-partition gets multiplied by  $D$ , due to each  $\lambda_m$  contributing with the power of  $D$  to  $\log |\text{diag}[\lambda^2]|$ . With these changes, the M-step in eqn. (20) finds its optimum when

$$D \frac{\tau_s}{2} \lambda_m^2 = \frac{\frac{t \cdot \delta t}{\tau_s} \langle \sum_{d=1}^D \mu_{j,m,d}^2 + \sigma_{j,m,d}^2 \rangle_j + \frac{\tau_s}{2} \nu_m \kappa_m^2}{\frac{2}{D} + \nu_m + \frac{t \cdot \delta t}{\tau_s}}. \quad (21)$$

Here, we have made a slight re-parameterization of  $\nu_m$  and  $\kappa_m$  to preserve the developed intuition that  $\kappa_m$  describes typical values of  $\lambda_m$ , and that  $\nu_m$  counts the number of pseudo-observations.<sup>3</sup>

## 3.2 Continuous-time, online inference

We now turn to a continuous-time formulation of the above motion structure inference algorithm. While doing so, we will overload the notation of time  $t$  which previously denoted integer-valued time steps and now becomes real-valued. We will point out the respective locations where this transition happens, below, to preclude confusion. Concretely, we will first reformulate the generative model in the form of natural parameters in [Section 3.2.1](#). Then, we derive continuous-time dynamics on these parameters in [Section 3.2.2](#), as had been promised in [Section 3.1.1](#), for solving the E-step. Finally, we cast eqn. (21) into a recursive equation for an online, continuous-time M-Step in [Section 3.2.3](#). This results in the reference *online EM algorithm* for online hierarchical motion structure inference.

### 3.2.1 Equivalent formulation of the generative model and inference using natural parameters

Knowing that all distributions involved in propagation, eqn. (7), and integration, eqn. (8), are multivariate Gaussians, we write the emission model, eqn. (4), in terms of the sufficient statistics of  $s_t$ ,

$$p(v_t | s_t) = \mathcal{N}(v_t; C s_t, \tilde{\sigma}_{\text{obs}}^2 I) \propto \exp \left[ \begin{pmatrix} s_t \\ s_t s_t^\top \end{pmatrix} \cdot \begin{pmatrix} \frac{C^\top v_t}{\tilde{\sigma}_{\text{obs}}^2} \\ -\frac{1}{2} \frac{C^\top C}{\tilde{\sigma}_{\text{obs}}^2} \end{pmatrix} \right], \quad (22)$$

where we have dropped all  $s_t$ -independent terms because they will play no role in the inference. Denoting the propagated, yet pre-integration, distribution by

$$p(s_t | v_{0:t-\delta t}) \propto \exp \left[ \begin{pmatrix} s_t \\ s_t s_t^\top \end{pmatrix} \cdot \begin{pmatrix} \Omega \mu_t \\ -\frac{1}{2} \Omega_t \end{pmatrix} \right], \quad (23)$$

with yet-to-be-determined natural parameters  $\Omega_t := \Sigma_t^{-1}$  and  $\Omega \mu_t := \Omega_t \mu_t$ , the integration of observations, eqn. (8), amounts to the following simple updates:

$$\Omega_t \mapsto \Omega_t + \frac{C^\top C}{\tilde{\sigma}_{\text{obs}}^2} \quad \text{and} \quad \Omega \mu_t \mapsto \Omega \mu_t + \frac{C^\top v_t}{\tilde{\sigma}_{\text{obs}}^2}. \quad (24)$$

In eqn. (23), we have made use already of the continuous-time notation, with time running from 0 to  $t$  and observations arriving at  $\delta t$ -intervals. We next address propagation, i.e., the continuous-time dynamics of eqn. (2) in terms of  $\Omega \mu_t$  and  $\Omega_t$ .

### 3.2.2 Continuous-time dynamics of natural parameters

For the OU process, eqn. (2), we know the evolution of the distribution of  $s_t$  between observations in closed form (see, e.g., (Gardiner, 2009)). Namely, the mean  $\mu_t$  decays towards  $\mathbf{0}$  exponentially with time constant  $\tau_s$ , and the covariance  $\Sigma_t$  decays towards its steady state value  $\text{diag}[\frac{\tau_s}{2} \lambda^2]$  exponentially with time constant  $\frac{\tau_s}{2}$ . From these known dynamics of  $\mu_t$  and  $\Sigma_t$ , we calculate the dynamics of the natural parameters:

$$\partial_t \Omega_t = -\Omega_t (\partial_t \Sigma_t) \Omega_t = \dots = \underbrace{\left( \frac{\tau_s}{2} \right)^{-1} (I - \Omega_t \text{diag}[\frac{\tau_s}{2} \lambda^2])}_{\otimes} \Omega_t + \frac{C^\top C}{\tilde{\sigma}_{\text{obs}}^2} \delta(t - t_{\text{obs}}). \quad (25)$$

<sup>3</sup>Specifically, we substituted  $\frac{\nu_m}{D} \rightarrow \nu_m$  and  $D \kappa_m^2 \rightarrow \kappa_m^2$ . Then,  $\nu_m = 1$  means one pseudo-observation in *each* spatial dimension. A uniform prior is imposed by  $\nu_m = -2/D$  and  $\kappa_m = 0$ .

For completeness, we have already included the integration of observations, eqn. (24), at observation time  $t_{\text{obs}}$  in the dynamics. In contrast to the dynamics of  $\Sigma_t$ , the dynamics of  $\Omega_t$  are non-linear. Yet, we observe that, in the absence of observations, eqn. (25) leads to the desired fixed point since  $\otimes = 0$  for  $\Omega^{-1} = \text{diag}[\frac{\tau_s}{2} \lambda^2]$ . Likewise, we obtain for the other natural parameter  $\Omega \mu_t$ :

$$\partial_t(\Omega \mu_t) = (\partial_t \Omega_t) \mu_t + \Omega_t (\partial_t \mu_t) = \dots = \tau_s^{-1} \underbrace{(I - 2\Omega_t \text{diag}[\frac{\tau_s}{2} \lambda^2])}_{\otimes \otimes} \Omega \mu_t + \frac{C^T v_t}{\sigma_{\text{obs}}^2} \delta(t - t_{\text{obs}}). \quad (26)$$

Again, as a sanity check, we observe the desired decay to zero because of  $\otimes \otimes \rightarrow -1$ , in the absence of observations.

**Continuous stream of observations.** So far, we have treated observations  $v_t$  as point observations which arrive only at distinct time points  $t_{\text{obs}}$  and, then, lead to “jumps” via integrating over the Dirac delta. For a complete continuous-time formulation, we choose to turn observations into a continuous input stream. When observations arrive every  $\delta t$ -interval and are corrupted by i.i.d. Gaussian noise of variance  $\tilde{\sigma}_{\text{obs}}^2$ , we can render their information content  $\delta t$ -independent by setting

$$\tilde{\sigma}_{\text{obs}}^2 = \sigma_{\text{obs}}^2 / \delta t \quad (27)$$

with the alternative parameter  $\sigma_{\text{obs}}^2$  being independent of the observation frame rate (see, e.g., (Jazwinski, 2007; Drugowitsch et al., 2012)). Furthermore, for  $\delta t \rightarrow 0$ , we use that

$$\delta t \delta(t - t_{\text{obs}}) \rightarrow 1 \quad (28)$$

because we get one Dirac delta-integration per  $\delta t$  in eqn. (25) and (26) while all other variables stay (almost) constant. With these two substitutions, we obtain:

$$\partial_t \Omega_t = \left(\frac{\tau_s}{2}\right)^{-1} (I - \Omega_t \text{diag}[\frac{\tau_s}{2} \lambda^2]) \Omega_t + \frac{C^T C}{\sigma_{\text{obs}}^2}, \quad (29)$$

$$\partial_t(\Omega \mu_t) = \tau_s^{-1} (I - 2\Omega_t \text{diag}[\frac{\tau_s}{2} \lambda^2]) \Omega \mu_t + \frac{C^T v_t}{\sigma_{\text{obs}}^2}, \quad (30)$$

with a continuous stream of observations,  $v_t$ . Together, eqn. (29) and (30) solve the E-step (from **Section 3.1.1**) by re-transforming the parameters via  $\Sigma_t = \Omega_t^{-1}$  and  $\mu_t = \Sigma_t \Omega \mu_t$ . These moments are used in the filtering posterior  $p(s_t | v_{0:t}; \lambda)$  of the reference online EM algorithm. The solution in terms of natural parameters is equivalent to the Kalman-Bucy filter (Kalman and Bucy, 1961) which is derived directly in terms of  $\mu_t$  and  $\Sigma_t$ , as we will see in **Section 3.3.1**.

### 3.2.3 Simultaneous online inference of motion sources and structure

We complete our derivation of online, hierarchical inference of motion sources and motion structure by casting the M-step, eqn. (20), into a recursive form and drawing the continuous-time limit. We restate eqn. (20) for reference,

$$\frac{\tau_s}{2} \lambda_m^2 = \frac{\frac{t \cdot \delta t}{\tau_s} \langle \mu_{j,m}^2 + \sigma_{j,m}^2 \rangle_j + \frac{\tau_s}{2} \nu_m \kappa_m^2}{2 + \nu_m + \frac{t \cdot \delta t}{\tau_s}}, \quad (31)$$

where  $\mu_{j,m}$  and  $\sigma_{j,m}^2 = \Sigma_{j,mm}$  are the posterior parameters obtained from the E-step. Recalling that  $\lambda_t^2$  is based on the time average of  $t$  discrete-time samples, we formulate  $\lambda_{t+1}^2$  as a sliding window average,

$$\lambda_{t+1}^2 = \left(1 - \frac{1}{t}\right) \cdot \lambda_t^2 + \frac{1}{t} \cdot \left(\frac{\tau_s}{2}\right)^{-1} \frac{\frac{t \cdot \delta t}{\tau_s} (\mu_{t+1}^2 + \sigma_{t+1}^2) + \frac{\tau_s}{2} \nu \kappa^2}{2 + \nu + \frac{t \cdot \delta t}{\tau_s}}, \quad (32)$$

where  $(1 - \frac{1}{t})$  and  $\frac{1}{t}$  are the weights for convex combination, and all vector operations are elementwise. Subtracting  $\lambda_t^2$  and dividing by inter-observation interval  $\delta t$ , we obtain:

$$\frac{\lambda_{t+1}^2 - \lambda_t^2}{\delta t} = -\frac{1}{t \delta t} \left( \lambda_t^2 - \left(\frac{\tau_s}{2}\right)^{-1} \frac{\frac{t \cdot \delta t}{\tau_s} (\mu_{t+1}^2 + \sigma_{t+1}^2) + \frac{\tau_s}{2} \nu \kappa^2}{2 + \nu + \frac{t \cdot \delta t}{\tau_s}} \right). \quad (33)$$

In this form, drawing the continuous-time limit is straight-forward. We let  $\delta t \rightarrow 0$  while keeping  $\tau_\lambda := t \delta t$  constant:

$$\partial_t \lambda_t^2 = -\frac{1}{\tau_\lambda} \left( \lambda_t^2 - \left( \frac{\tau_s}{2} \right)^{-1} \frac{\frac{\tau_\lambda}{\tau_s} (\mu_t^2 + \sigma_t^2) + \frac{\tau_s}{2} \nu \kappa^2}{2 + \nu + \frac{\tau_\lambda}{\tau_s}} \right). \quad (34)$$

Time  $t$  is now in continuous-time. The time constant  $\tau_\lambda$  is the (continuous-time) width of the integration window and defines the minimum time scale at which significant changes of  $\lambda_t^2$  are expected to occur. From a strict algorithmic perspective of EM, we require that  $\tau_\lambda \gg \tau_s$ . However, we observe in computer simulations that for practical applications even small values,  $\tau_\lambda \gtrsim \tau_s$ , work reliably.

**Extension to multiple spatial dimensions.** We conclude by generalizing eqn. (34) to multiple spatial dimensions via comparison of eqn. (20) and eqn. (21):

$$\partial_t \lambda_t^2 = -\frac{1}{\tau_\lambda} \left( \lambda_t^2 - \left( \frac{D \tau_s}{2} \right)^{-1} \frac{\frac{\tau_\lambda}{\tau_s} (\sum_{d=1}^D \mu_{t,d}^2 + \sigma_{t,d}^2) + \frac{\tau_s}{2} \nu \kappa^2}{\frac{2}{D} + \nu + \frac{\tau_\lambda}{\tau_s}} \right). \quad (35)$$

Eqn. (35) is used for inferring the motion strengths  $\lambda_t^2$  in the reference online EM algorithm. The interactions of eqn. (29), (30) and (35) confirm and particularize our earlier chicken-and-egg intuition that motion sources,  $s_t$ , and motion structure,  $\lambda_t$ , are mutually coupled. The exact interactions are rather convoluted, and we will identify in **Section 3.3** an approximate interaction that is intuitively more accessible.

### 3.3 Adiabatic approximation for prediction error-based inference

The inference algorithm described by eqn. (29), (30), and (35) is a nice solution from a mathematical perspective. Yet, for a theory of brain computation, it is questionable whether neural dynamics could reliably calculate 3rd-order polynomials in the variables  $\Omega \mu_t$ ,  $\Omega_t$ , and  $\lambda_t^2$  as demanded, for instance, by eqn. (30).

In the following, we therefore explore an alternative, approximate solution which, as we will see, considerably simplifies the involved computations while leading to almost identical results during motion structure inference. We will proceed in three steps. First, in **Section 3.3.1**, we transfer eqn. (29) + (30) back into the domain of moments,  $\mu_t$  and  $\Sigma_t$ , and recover a prediction error-based update equation known as the Kalman-Bucy filter (Kalman and Bucy, 1961; Jazwinski, 2007). Then, in **Section 3.3.2**, we introduce an adiabatic approximation for the posterior covariance,  $\Sigma_t$ , assuming that it has always converged to stationarity. Finally, in **Section 3.3.3**, we derive an analytical solution for the converged uncertainty for the special case of diagonal covariance. The resulting algorithm, which we term the *adiabatic observer model* and which is the online algorithm used in the main text, features interpretable dynamics on the behaviorally relevant quantities  $\mu_t$  and  $\lambda_t^2$  and relies on only quadratic computations of vector-valued variables, rather than 3rd-order computations on matrix-valued variables.

#### 3.3.1 Prediction error-based formulation

We first transform eqn. (29) into dynamics of  $\Sigma_t$ :

$$\partial_t \Sigma_t = -\Sigma_t (\partial_t \Omega_t) \Sigma_t \quad (36)$$

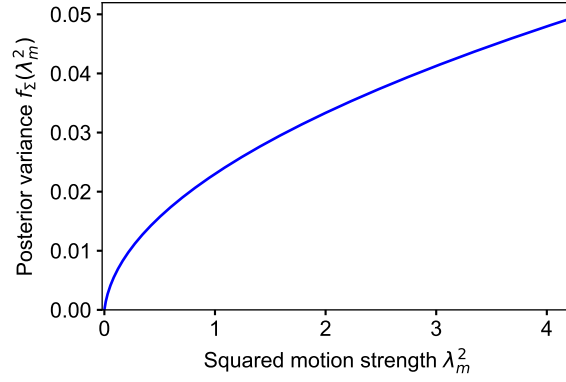
$$= -\left( \frac{\tau_s}{2} \right)^{-1} \Sigma_t + \text{diag}[\lambda^2] - \Sigma_t \frac{C^\top C}{\sigma_{\text{obs}}^2} \Sigma_t, \quad (37)$$

and use this result for transforming eqn. (30) into dynamics of  $\mu_t$ :

$$\partial_t \mu_t = \partial_t (\Sigma_t \Omega \mu_t) = (\partial_t \Sigma_t) \Omega \mu_t + \Sigma_t \partial_t \Omega \mu_t \quad (38)$$

$$= -\frac{\mu_t}{\tau_s} + \Sigma_t C^\top \left( \frac{v_t}{\sigma_{\text{obs}}^2} - \frac{C \mu_t}{\sigma_{\text{obs}}^2} \right). \quad (39)$$

Eqn. (39) is a pretty neat equation as it reveals the “inner working” of inference as updating  $\mu_t$  with the help of prediction errors  $(v_t - C \mu_t)$ , which are projected “up” into motion source space via  $C^\top$ , and then weighted by the relative uncertainty of internal estimates vs. the uncertainty of observations,  $\Sigma_t / \sigma_{\text{obs}}^2$ . In particular, in the absence of observations ( $\sigma_{\text{obs}}^2 \rightarrow \infty$ ), the estimated mean  $\mu_t = \langle s_t \rangle$  decays to zero with time constant  $\tau_s$ , as expected from the OU process. At first glance, it may seem that the inferred structure,  $\lambda^2$ , plays no role in inferring  $s_t$  anymore. But actually,  $\lambda^2$  still is present in eqn. (39) indirectly through its effect on  $\Sigma_t$ . We will study this indirect effect in the following.



**Figure S7. Adiabatic, diagonal solution for the posterior variance.** Shown is the function  $f_{\Sigma}(\lambda_m^2)$ , given by eqn. (42). Parameters:  $\tau_s = 300$  ms,  $\sigma_{\text{obs}} = 0.05$ ,  $\|c_m\|^2 = 4$ .

### 3.3.2 Convergence approximation on the posterior precision

For  $\tau_{\lambda} > \tau_s$ , we observe that  $\Sigma_t$  can be calculated directly as a function of  $\lambda_t^2$ , instead of going through the hassle of integrating eqn. (37). This can be seen as follows. We know that the posterior covariance  $\Sigma_t$  decays towards its stationary value with a time constant in the order of  $\tau_s/2$ . The stationary value itself is a dynamic equilibrium between increasing uncertainty due to diffusion (the underlying Wiener process in eqn. (2)) and decreasing uncertainty due to incoming observations (corresponds to the term  $C^T C / \sigma_{\text{obs}}^2$  in eqn. (29)). Notably, the stationary value does not depend on the observations,  $v_t$ . This is a peculiarity of the inference task at hand which is known from Kalman filtering. This leaves  $\lambda_t^2$  as the only dynamic variable in eqn. (37) to influence the stationary point of  $\Sigma_t$ . Since  $\lambda_t^2$  changes on time scales  $\tau_{\lambda} > \tau_s$ , the covariance,  $\Sigma_t$ , will always have enough time to react to any change in its stationary value. This justifies treating  $\Sigma_t$  as having converged at any time, a method known in physics as adiabatic approximation.

For stationary  $\Sigma_t$ , it follows from setting  $\partial_t \Sigma_t = 0$  in eqn. (37) that

$$\frac{\tau_s}{2} \Sigma \frac{C^T C}{\sigma_{\text{obs}}^2} \Sigma + \Sigma - \text{diag} \left[ \frac{\tau_s}{2} \lambda^2 \right] = 0. \quad (40)$$

### 3.3.3 Analytic solution for diagonal covariance matrices

Eqn. (40) is a continuous-time algebraic Riccati equation which can, in general, be solved using eigendecompositions of an extended matrix. However, for a neural implementation, we will follow a simpler route by assuming that  $\Sigma$  is diagonal. This amounts to ignoring correlations in uncertainty about latent motion sources in the posterior distribution, for instance, during reasoning of the type: “I have correctly decomposed all velocities in expectation, and I know my uncertainty about each motion component. But if I underestimated the flock velocity, then I likely overestimated the birds’ individual velocities.” Only the last step of this reasoning will be ignored by dropping off-diagonal elements in  $\Sigma$ . We observe in computer simulations that neglecting these posterior correlations typically has little impact on the solution.

For diagonal  $\Sigma$ , eqn. (40) can be solved for each element  $\sigma_m^2 := \Sigma_{mm}$  separately:

$$\frac{\tau_s}{2} \frac{\|c_m\|^2}{\sigma_{\text{obs}}^2} (\sigma_m^2)^2 + \sigma_m^2 - \frac{\tau_s}{2} \lambda_m^2 = 0, \quad (41)$$

where we have defined  $\|c_m\|^2 = \sum_{k=1}^K C_{km}^2$  to denote the vector-norm of the  $m$ -th column of  $C$ , that is, the squared Euclidean length of the  $m$ -th motion component. Solving eqn. (41) is straightforward, and we denote with  $f_{\Sigma}(\lambda_m^2)$  the resulting function for calculating  $\sigma_m^2$  as a function of  $\lambda_m^2$ :

$$\sigma_m^2 = f_{\Sigma}(\lambda_m^2) = \frac{\sigma_{\text{obs}}^2}{\tau_s \|c_m\|^2} \left( -1 + \sqrt{1 + \frac{\tau_s^2 \|c_m\|^2}{\sigma_{\text{obs}}^2} \lambda_m^2} \right). \quad (42)$$

$f_{\Sigma}$  is a monotonically increasing, non-negative function. Its graph is shown in **Fig. S7** for typical parameter values. In the limit of small motion strengths,  $\lambda_m \rightarrow 0$ , the variance grows quadratically in  $\lambda_m$  (non-squared):  $f_{\Sigma}(\lambda_m^2) \approx \frac{\tau_s}{2} \lambda_m^2$ . For large strengths,  $\lambda_m \rightarrow \infty$ , the variance becomes linear in  $\lambda_m$ :  $f_{\Sigma}(\lambda_m^2) \approx \frac{\sigma_{\text{obs}}}{\|c_m\|} \lambda_m$ .



### 3.3.4 Putting it together: neuro-friendly algorithm for online structure inference

To obtain the neuro-friendly *adiabatic observer model*, we simply plug eqn. (42) into the dynamics of  $\lambda_t^2$ , given by eqn. (35), and  $\mu_t$ , given by eqn. (39):

$$\partial_t \lambda_t^2 = -\frac{1}{\tau_\lambda} \left( \lambda_t^2 - \left( \frac{D \tau_s}{2} \right)^{-1} \frac{\tau_\lambda}{\tau_s} \left( \sum_{d=1}^D \mu_{t,d}^2 + f_\Sigma(\lambda_t^2) \right) + \frac{\tau_s}{2} \nu \kappa^2 \right), \quad (43)$$

$$\partial_t \mu_t = -\frac{\mu_t}{\tau_s} + f_\Sigma(\lambda_t^2) \mathbf{C}^\top \left( \frac{v_t}{\sigma_{\text{obs}}^2} - \frac{\mathbf{C} \mu_t}{\sigma_{\text{obs}}^2} \right), \quad (44)$$

$$\text{with } f_\Sigma(\lambda_m^2) = \frac{\sigma_{\text{obs}}^2}{\tau_s \|\mathbf{c}_m\|^2} \left( -1 + \sqrt{1 + \frac{\tau_s^2 \|\mathbf{c}_m\|^2}{\sigma_{\text{obs}}^2} \lambda_m^2} \right).$$

In this vector notation,  $f_\Sigma(\lambda_t^2)$  is evaluated elementwise, and the sum in eqn. (43) includes one evaluation of  $f_\Sigma$  for each spatial dimension  $d = 1..D$ . In eqn. (44),  $f_\Sigma(\lambda_t^2)$  is multiplied elementwise with the ‘‘up-projected’’ prediction error.

We recognize how the motion structure,  $\lambda_t$ , controls the gating function,  $f_\Sigma(\lambda_t^2)$ , for performing the credit assignment of the prediction errors, in eqn. (44). Furthermore, we note that also the posterior uncertainty can be recovered at any time since  $\sigma_{t,m}^2 = f_\Sigma(\lambda_{t,m}^2)$ .

For  $D = 1$ , eqn. (43) is eqn. (1) from the main text, eqn. (44) is eqn. (2), and eqn. (42) is eqn. (3).

### 3.3.5 A pleasant note on inference of rotational and radial motion

The approximations introduced for the adiabatic observer model hold, remarkably and importantly, also for the biologically relevant case of radial and rotational motion (cf. **Section 2.5** and **Fig. S6** for the generative model).

To illustrate this, consider the case of two motion sources, radial  $s_{\text{rad}}(t)$  and rotational  $s_{\text{rot}}(t)$ , with motion features given by eqn. (6). The motion features  $\mathbf{C}$  depend on the receptive field locations with parameters  $R_k$  and  $\theta_k$ . For inferring the radial component,  $s_{\text{rad}}(t)$ , all receptive fields are equally informative, irrespective of their eccentricity,  $R_k$ . Accordingly, eqn. (44) weighs all radial prediction errors equally, as expressed by the  $R_k$ -independence of the radial row in  $\mathbf{C}^\top$ . This is different when estimating rotational motion,  $s_{\text{rot}}(t)$ . Here, prediction errors measured near the fovea (small  $R_k$ ) contribute only little to  $\partial_t \mu_{\text{rot}}$ : for small  $R_k$ , we expect to observe a small rotational motion energy via  $\mathbf{C} \mu_t$ , such that the ‘‘Cartesian’’ observation noise of size  $\sigma_{\text{obs}}$  makes the stimulus virtually uninformative about the rotational velocity  $\dot{\phi}$ . The scaling of  $\mathbf{C}^\top$  with  $R_k$  accounts for that. Receptive fields far away from the fovea, in contrast, predict a strong rotational velocity via  $\mathbf{C} \mu_t$ , such that noise of size  $\sigma_{\text{obs}}$  (in Cartesian space) has only a minor impact on estimating  $\mu_{\text{rot}}$ . Accordingly, the scaling of  $\mathbf{C}^\top$  with  $R_k$  assigns a higher weight to peripheral receptive fields for estimating rotational motion.

As a final remark, the above example assumed the observation noise  $\sigma_{\text{obs}}$  to be constant across all receptive fields. Eqn. (44) naturally supports extensions to heterogeneous observation noise because all local prediction errors are measured in units of their local noise.

## 4. Extensions of the online algorithm

### 4.1 Non-isotropic observation noise and missing observations

For the main manuscript, we have assumed the observation noise,  $\sigma_{\text{obs}}$ , to be a constant across time,  $t$ , observed features,  $k$ , and spatial dimensions,  $d$ . In real-world scenes, the observation noise could change along all those indices. An object could be occluded, or otherwise temporarily invisible, leading to  $1/\sigma_{\text{obs}}^2 = 0$ . Different objects might have different observation noise, e.g., due to different visual contrast. The aperture problem could render local velocity signals ambiguous: e.g., what is the direction of motion for a straight line that is larger than the aperture? This could be modeled by small observation noise perpendicular to the line, and large noise parallel to the line. The so-constructed diagonal covariance matrix,  $\Sigma_{\text{diag}}$ , is then rotated as per  $\Sigma_{xy} = \mathbf{Q}^\top \Sigma_{\text{diag}} \mathbf{Q}$  with rotation matrix,  $\mathbf{Q}$ , into the canonical  $x$ - $y$ -coordinate frame.

The above extensions of the observation noise are supported by our online algorithm, as long as changes in  $\sigma_{\text{obs}}^2$  occur slower than  $\tau_s$  (so the adiabatic approximation remains valid). In eqn. (44),  $\frac{1}{\sigma_{\text{obs}}^2}$  is extended to have different

elements,  $\frac{1}{\sigma_{\text{obs},k,d}^2}$ , and is multiplied elementwise with  $\mathbf{v}_{t,k,d}$  and  $(\mathbf{C}\boldsymbol{\mu})_{t,k,d}$ . When calculating  $f_{\Sigma}$ , replace

$$\frac{\|\mathbf{c}_m\|^2}{\sigma_{\text{obs}}^2} \quad \text{by} \quad \left( \mathbf{C}^\top \text{diag} \left[ \frac{\mathbf{1}}{\sigma_{\text{obs}}^2} \right] \mathbf{C} \right)_{mmm} \quad (45)$$

for each spatial dimension. In eqn. (43), the summation then runs over the different spatial dimensions of  $f_{\Sigma}$ . The idea that  $\frac{1}{\sigma_{\text{obs}}^2}$  is vector-valued is also used in the network implementation in **Section 5**.

A note on the provided Python code package: In the code for the network,  $\sigma_{\text{obs}}^2$  can currently have different values for every input velocity  $v_k$ , but is assumed to be (a) identical in both spatial dimensions, and (b) not to change over time. The code for the algorithm is less restrictive by supporting temporary masking of inputs (presented via `class ObservationGeneratorVelo`; leading to posterior variance  $\sigma_m^2 = f_{\Sigma}(\lambda_m^2) = \frac{\tau_s}{2} \lambda_m^2$  if all dependent objects are invisible according to eqn. (41)) and different noise in spatial dimensions.

## 4.2 Heterogeneous time constants

Different motion components,  $s_m$ , might have different time-constants,  $\tau_s$ , for typical changes in speed and direction to occur. An extension to a vector  $\boldsymbol{\tau}_s = (\tau_{s,1}, \dots, \tau_{s,M})$  is straightforward by replacing all occurrences of  $\tau_s$  and  $1/\tau_s$  by  $\text{diag}[\boldsymbol{\tau}_s]$  and  $\text{diag}[\boldsymbol{\tau}_s^{-1}]$ , respectively. The code package supports heterogeneous time constants for both the algorithm and the network.

## 4.3 Interaction priors capturing feature compatibility

Some motion components may be unlikely to occur together. Consider, for example, two cluster components in  $\mathbf{C}$  which are overlapping but do not contain one another: the simultaneous occurrence of the two clusters would not be compatible with a tree structure. We can accommodate such incompatibility with the help of an *interaction prior*.

In the following, we outline how interaction priors can be included in the theory, and how they will affect the inference process. We endow the  $\lambda^2$ -prior from eqn. (16) with an interaction term:

$$p(\lambda^2; \boldsymbol{\nu}, \boldsymbol{\kappa}^2) \propto \left[ \prod_{m=1}^M \mathcal{I}\chi(\lambda_m^2; \nu_m, \kappa_m^2) \right] \cdot e^{-\frac{1}{2}(\lambda^2)^\top \mathbf{J}(\lambda^2)}, \quad (46)$$

where the interaction matrix  $\mathbf{J} \in \mathbb{R}^{M \times M}$  is a symmetric, zero-diagonal matrix that models feature incompatibility. For instance, positive values,  $J_{ml} = J_{lm} > 0$ , describe a (soft) incompatibility between the  $m^{\text{th}}$  and  $l^{\text{th}}$  motion component.

In the derivation of the M-step, the interaction prior leads to an additional term  $(-\mathbf{J}\boldsymbol{\lambda})_m$  on the right-hand side of eqn. (17).<sup>4</sup> For the optimum in eqn. (19), this leads to the following equation (using vector notation and covering multiple spatial dimensions):

$$D \frac{\tau_s}{2} \left( \mathbf{I} + \frac{2}{2/D + \boldsymbol{\nu} + t} \text{diag}[(\lambda^2)^2] \mathbf{J} \right) \boldsymbol{\lambda}^2 = \frac{t \sum_d \langle \boldsymbol{\mu}_j^2 + \sigma_j^2 \rangle_j + \frac{\tau_s}{2} \boldsymbol{\nu} \boldsymbol{\kappa}^2}{2/D + \boldsymbol{\nu} + t}. \quad (47)$$

For small values of  $\|\mathbf{J}\|$  or, similarly, large values of  $t$  ( $=\tau_\lambda/\tau_s$  in continuous-time), the matrix  $\tilde{\mathbf{J}} := \left( \mathbf{I} + \frac{2}{2/D + \boldsymbol{\nu} + t} \text{diag}[(\lambda^2)^2] \mathbf{J} \right)$  is invertible with the approximate inverse  $\tilde{\mathbf{J}}^{-1} \approx \left( \mathbf{I} - \frac{2}{2/D + \boldsymbol{\nu} + t} \text{diag}[(\lambda^2)^2] \mathbf{J} \right)$ . We can therefore move this matrix to the right-hand side and follow the derivation for online inference without interaction priors. This leads to the following equivalent of eqn. (43):

$$\partial_t \lambda_t^2 = -\frac{1}{\tau_\lambda} \left( \lambda_t^2 - \left( \frac{D \tau_s}{2} \right)^{-1} \left( \mathbf{I} - \frac{2}{\frac{2}{D} + \boldsymbol{\nu} + \frac{\tau_\lambda}{\tau_s}} \text{diag}[(\lambda^2)^2] \mathbf{J} \right) \frac{\frac{\tau_\lambda}{\tau_s} (\sum_{d=1}^D \boldsymbol{\mu}_{t,d}^2 + f_{\Sigma}(\lambda_t^2)) + \frac{\tau_s}{2} \boldsymbol{\nu} \boldsymbol{\kappa}^2}{\frac{2}{D} + \boldsymbol{\nu} + \frac{\tau_\lambda}{\tau_s}} \right). \quad (48)$$

The only difference to eqn. (43) is that the target values on the right hand side are mixed together via  $\tilde{\mathbf{J}}^{-1}$ . This gives rise to quite intuitive dynamics: if two motion components are incompatible, they mutually subtract their respective (independent) target values from another, thereby slightly changing the motion structure in which the E-step will interpret future input and, ultimately, leading to soft winner-takes-all competition. The term  $(\text{diag}[(\lambda^2)^2])$  limits the competition to those components that are significantly different from zero.

<sup>4</sup>Note that the normalization of the distribution in eqn. (46) will not depend on  $\lambda^2$ , and thus will play no role in the M-step, which uses  $\lambda^2$ -derivatives of  $\log p(\lambda^2)$ .

#### 4.4 Detecting motion components that had decayed to baseline

If motion components are not detected in the structure for a longer time, the associated strength,  $\lambda_m$ , will decay to zero. Since  $f_\Sigma(0)=0$ , this will prohibit future detection of said motion component. This issue can be addressed in two ways. In a biological agent, noise in the nervous system will lead to small fluctuations in the encoded value of  $\lambda$ , thereby “probing” the presence of motion components simply via noisy deviations from  $\lambda_m=0$ . Alternatively, a more principled solution exploits the hyperparameters,  $\nu_m$  and  $\kappa_m$ , in eqn. (16) to prevent  $\lambda_m$  from decaying to zero, e.g., by choosing  $\nu_m=1$  and  $\kappa_m=0.1$ . This follows the intuition that pseudo-observations in support of  $\lambda_m=\kappa_m$  had been observed for a duration  $\nu_m \tau_s$ .

#### 4.5 Learning the motion components on long time-scales

While not being the focus of this work, we briefly touch upon the question of how the motion components,  $C$ , could be learned online from observations in an unsupervised manner. To this end, we follow a similar EM scheme as for inferring  $\lambda$  and note that in eqn. (13) only the quadratic term depends on  $C$ . For the M-step, however, instead of maximizing  $Q$  directly, we perform gradient ascent with respect to  $C$ :

$$\nabla_C Q_t(C) = \langle \nabla_C [-\frac{1}{2\sigma_{\text{obs}}^2} (\mathbf{v}_t - C\mathbf{s}_t)^\top (\mathbf{v}_t - C\mathbf{s}_t)] \rangle_{p(\mathbf{s}_t | \mathbf{v}_{0:t})} \quad (49)$$

$$= \frac{1}{\sigma_{\text{obs}}^2} \langle \mathbf{v}_t \mathbf{s}_t^\top - C (\mathbf{s}_t \mathbf{s}_t^\top) \rangle_{p(\mathbf{s}_t | \mathbf{v}_{0:t})} \quad (50)$$

$$= \frac{1}{\sigma_{\text{obs}}^2} \left( \mathbf{v}_t \boldsymbol{\mu}_t^\top - C \left( \boldsymbol{\mu}_t \boldsymbol{\mu}_t^\top + \boldsymbol{\Sigma}_t \right) \right). \quad (51)$$

This gradient establishes the intuition to compare the observed covariance between inputs and motion components against their expected covariance.

Furthermore, we note that the parameterization of motion structure via  $C$  and  $\lambda$  leaves an invariance: any scaling of  $\lambda$  can be compensated by an inverse scaling of  $C$ . Due to the sparsity prior on  $\lambda$ , which favors small values, this bears the risk of unbounded growth in  $C$ . We can address this risk by imposing a regularizing prior on  $C$ , e.g., a Laplace prior or a Gaussian prior, such that the system finds a balance between small  $\lambda$  and small  $\|C\|$ . We incorporate the regularizer in the gradient-based update rule with the aim to balance prior and likelihood such that the likelihood is weighted to contribute  $N_C$  independent samples:

$$\partial_t C = \eta_C \left[ \frac{1}{\sigma_{\text{obs}}^2} \left( \mathbf{v}_t \boldsymbol{\mu}_t^\top - C \left( \boldsymbol{\mu}_t \boldsymbol{\mu}_t^\top + \boldsymbol{\Sigma}_t \right) \right) + \frac{1}{N_C} \nabla_C \log p(C) \right]. \quad (52)$$

Examples for the regularizer are  $\nabla_C \log p(C) = -\text{sign}(C)/b$  for a Laplace prior and  $\nabla_C \log p(C) = -C/b$  for a Gaussian prior. For the online M-step, both  $\boldsymbol{\mu}_t$  and  $\lambda_t$  must have time to adapt to any changes in  $C$ . This is ensured by requiring that the learning rate,  $\eta_C$ , is small enough to average over a large number of independent samples, i.e.,  $\eta_C \ll 1/\tau_\lambda$ . The small learning rate emphasizes how we think of  $C$ -optimization as a long-term learning process of a “feature dictionary”.

Eqn. (52) offers a path to learning  $C$  via an online EM algorithm. In a network implementation with linear population codes, however, it remains unclear how the update could be communicated to synapses:  $C$  determines many synaptic weights in the network. Finding encoding and decoding vectors of the variables to support simple plasticity rules is future work.

## 5. Neural network implementation

How could biological recurrent neural networks implement online motion structure inference? In light of the theory developed in **Section 3.3**, we will operationalize this question by implementing eqn. (43) and eqn. (44).

While we will strive to incorporate salient properties from motion sensitive brain areas, the exact computational mechanisms underlying many experimental findings are still elusive. Thus, inevitably, several modeling assumptions have to be made. These assumptions are presented in **Section 5.1**. We then discuss, in **Section 5.2**, which variables—input and latent—we choose to be linearly decodable by downstream populations, and, in **Section 5.3**, express the adiabatic observer model in terms of these variables. For performing the required computations on these variables, we will extend the ideas developed in (Beck, Latham, and Pouget, 2011), in **Section 5.4**, to a systematic theory of neural integration of high-dimensional linear and quadratic differential equations. In **Section 5.5**, we will apply the theory to derive a rate-based recurrent neural network model for online motion structure inference. Finally, in **Section 5.6**, we introduce—as an example for the computer simulations—a neural encoding model of input variables that captures many properties of middle temporal visual area (MT) while staying mathematically tractable with regard to its computational function.

### 5.1 Aims and assumptions

We view our network model as a starting point for an experiment–theory loop. Some neural response properties will be rather general and could be tested in experiments directly. Others will be more specific and could guide targeted experiments. In any case, we expect that many aspects of this initial model will be revised and refined in the process.

For the model, we make three assumptions:

- **Rate-based network.** We assume that all information is conveyed in the neuronal firing rates. Thus, no exact spike-timing is considered. Further, we will allow negative firing rates—think of them as negative deviations from a baseline value.
- **Linear and quadratic operations.** We assume that neurons can integrate their synaptic inputs in two ways: linearly and quadratically. Specifically, we assume that the dynamics of the firing rate of a neuron (or small population)  $i$  takes the form,

$$\tau_i \partial_t r_i = -r_i + f_i(\mathbf{w}_i^\top \mathbf{r} + \mathbf{r}^\top \mathbf{Q}^{(i)} \mathbf{r} + b_i) , \quad (53)$$

with time constant  $\tau_i$ , a potentially non-linear activation function  $f_i$ , weight vector  $\mathbf{w}_i^\top$ , quadratic interaction matrix  $\mathbf{Q}^{(i)}$ , and bias  $b_i$ . In the main text, we had omitted the (per-neuron) bias,  $b_i$ , because it can be absorbed in  $f_i$ . For the following formal derivation, we make the bias explicit for clarity, and, as we will see, this leads to activation functions,  $f_i$ , which are different only on a per-population basis. Eqn. (53) is a standard form for rate-based network models (Dayan and Abbott, 2001), and quadratic interactions are commonly used in neural network modeling (Salinas and Abbott, 1996; Beck, Latham, and Pouget, 2011).

- **Linear decoding of task-relevant variables.** A subset of variables, especially those which are relevant for actions and decision making, can be read out linearly by downstream populations. We will specify the subset of variables in the next section.

### 5.2 Linearly decodable variables

Inspecting eqn. (43) and eqn. (44), an elegant decomposition into basic operations (that is, addition, linear and quadratic multiplication) employs the following variables:

$$\mu_t, \quad \lambda_t^2, \quad \frac{v_t}{\sigma_{\text{obs}}^2}, \quad \frac{1}{\sigma_{\text{obs}}^2}, \quad \underbrace{\left( \frac{v_t}{\sigma_{\text{obs}}^2} - \frac{\mathbf{C} \mu_t}{\sigma_{\text{obs}}^2} \right)}_{\text{Pred. err. } \epsilon_t}, \quad f_\Sigma(\lambda_t^2) . \quad (54)$$

The first three variables,  $\mu_t$ ,  $\lambda_t^2$ ,  $v_t$ , are directly related to the task of decomposing visual scenes. Further, we include the observation noise,  $1/\sigma_{\text{obs}}^2$ , as an input variable (rather than treating it as a constant) to accommodate the extended theory presented in **Section 4.1**, permitting the network to handle, for instance, transient occlusion of objects. Of course, the value of  $\sigma_{\text{obs}}^2$  could also be a constant in the network. The prediction error,  $\epsilon_t$ , is an auxiliary variable to restrict the complexity of operations to being at most quadratic. (We will see in **Section 5.4** that the computational complexity is directly inherited by the neural dynamics.) Finally, the posterior variance,  $f_\Sigma(\lambda_t^2)$ , which is a hallmark

of Bayesian computation, is required for motion structure decomposition and, potentially, for Bayesian decision making.

The variables listed in eqn. (54) are assumed to be linearly decodable, that is, they can be read out from neural activity (at the example of  $\mu_t$ ) via

$$\mu_t = A^\mu r_t , \quad (55)$$

with some readout matrix  $A^\mu$ . The other variables maintain corresponding matrices  $A^\lambda$ ,  $A^v$ ,  $A^\sigma$ ,  $A^\epsilon$ , and  $A^\Sigma$ , respectively. Here,  $r_t$  are the instantaneous firing rates of a population of neurons that encode  $\mu_t$ . Multiple variables can be encoded by the same neural population.

### 5.3 Motion structure inference via at-most quadratic operations

First, we observe that eqn. (43) and eqn. (44) almost exclusively contain linear and quadratic terms when expressed in the variables of eqn. (54):

$$\partial_t \lambda_t^2 = -\frac{1}{\tau_\lambda} \lambda_t^2 + \frac{2}{D \tau_s \tau_\lambda \left(\frac{2}{D} + \nu + \frac{\tau_\lambda}{\tau_s}\right)} \left( \frac{\tau_\lambda}{\tau_s} \sum_{d=1}^D \mu_{t,d} \odot \mu_{t,d} + \frac{D \tau_\lambda}{\tau_s} f_\Sigma(\lambda_t^2) + \frac{\tau_s}{2} \nu \kappa^2 \right) , \quad (56)$$

$$\partial_t \mu_t = -\frac{1}{\tau_s} \mu_t + f_\Sigma(\lambda_t^2) \odot C^\top \epsilon_t , \quad (57)$$

$$\partial_t \epsilon_t = -\frac{1}{\tau_\epsilon} \left( \epsilon_t - \frac{v_t}{\sigma_{\text{obs}}^2} + \frac{1}{\sigma_{\text{obs}}^2} \odot C \mu_t \right) = -\frac{1}{\tau_\epsilon} \epsilon_t + \frac{1}{\tau_\epsilon} \frac{v_t}{\sigma_{\text{obs}}^2} - \frac{1}{\tau_\epsilon} \frac{1}{\sigma_{\text{obs}}^2} \odot C \mu_t , \quad (58)$$

where we have made elementwise multiplication explicit via the  $\odot$ -operator, and moved the prediction error,  $\epsilon_t$ , into a separate dynamic equation with time constant  $\tau_\epsilon$ . For this separation to maintain faithful results, we require that  $\tau_\epsilon < \tau_s$ , such that the prediction error can react to changes in  $\mu_t$ . The only variable in eqn. (54) that cannot be calculated within this scheme is  $f_\Sigma(\lambda_t^2)$ , which contains a square root, and, thus, has to be addressed separately, below. The variables  $v_t/\sigma_{\text{obs}}^2$  and  $1/\sigma_{\text{obs}}^2$  are the input variables that are fed into the system.

### 5.4 Neural dynamics for integrating linear and quadratic differential equations

We will now establish how linear and quadratic dynamics of latent variables, such as eqn. (56)–(58), can be integrated in neural space. What follows is basically a clearly structured generalization of (Beck, Latham, and Pouget, 2011).

**Notation in the algorithmic domain.** Inevitably, some notation has to be introduced for addressing all of the above dynamics in both the *algorithmic domain* (i.e., dynamics of variables) and the *network domain* (i.e., dynamics of neuronal firing rates). To keep the presentation general, we will adopt variable-dynamics of the generic form

$$\partial_t z = \mathbf{y} \mathbf{Q} \mathbf{x} + \mathbf{W} \mathbf{x} + \mathbf{b} \quad \text{with} \quad (\mathbf{y} \mathbf{Q} \mathbf{x})_i \stackrel{\text{def}}{=} \sum_{j,k} Q_{ijk} y_j x_k . \quad (59)$$

Here,  $\mathbf{z}$ ,  $\mathbf{y}$ , and  $\mathbf{x}$  denote vector-valued variables. Further,  $\mathbf{b}$  is a vector-valued additive constant,  $\mathbf{W}$  a matrix, and  $\mathbf{Q}$  a 3rd-order tensor, for which we establish the notation of small, capital, and underlined capital letters, respectively.

Note that expressions with elementwise multiplication are covered by the tensors. For instance, using Einstein summation convention,

$$(\mathbf{y} \odot \mathbf{x})_i = y_i x_i = \delta_{ij} \delta_{ik} y_j x_k = (\mathbf{y} \mathbf{Q} \mathbf{x})_i \quad \text{with} \quad Q_{ijk} = \delta_{ij} \delta_{ik} , \quad (60)$$

$$[\mathbf{W}^1 (\mathbf{y} \odot \mathbf{W}^2 \mathbf{x})]_i = W_{ij}^1 (\mathbf{y} \odot \mathbf{W}^2 \mathbf{x})_j = W_{ij}^1 (y_j W_{jk}^2 x_k) = (\mathbf{y} \mathbf{Q} \mathbf{x})_i \quad \text{with} \quad Q_{ijk} = W_{ij}^1 W_{jk}^2 , \quad (61)$$

$$[(\mathbf{W}^1 \mathbf{y}) \odot (\mathbf{W}^2 \mathbf{x})]_i = W_{ij}^1 y_j W_{ik}^2 x_k = (\mathbf{y} \mathbf{Q} \mathbf{x})_i \quad \text{with} \quad Q_{ijk} = W_{ij}^1 W_{ik}^2 . \quad (62)$$

Thus, all algorithmic dynamics in eqn. (56)–(58) are of form eqn. (59).

**Notation in the network domain.** We now turn to the question of how neuronal populations can calculate dynamics of the form in eqn. (59) when the involved variables are linearly decodable, that is, when  $\mathbf{z} = A^z r^z$ . Refining the notation in eqn. (55), we will make explicit which population  $r^z$  encodes variable  $\mathbf{z}$  and suppress the time dependence in  $r_t^z$ . Again, we emphasize that differently denoted populations, e.g.,  $r^z$  and  $r^x$ , can and often will refer to the same population—the refined notation simply gives us the flexibility to cover various cases.

Following (Beck, Latham, and Pouget, 2011), we will further make use of what is called the *adjoint matrix*  $A^{z\dagger}$  of matrix  $A^z$ , which is characterized by  $A^z A^{z\dagger} = I$ . Such right-inverse, albeit not unique, always exists if the rows of  $A^z$  are linearly independent, which is commonly fulfilled when the number of neurons exceeds the number of variables. If variables  $z$  and  $x$  are encoded by the same populations of neurons, we further require that  $A^z A^{x\dagger} = A^x A^{z\dagger} = \mathbf{0}$ . As long as these orthogonality conditions are satisfied, the exact form of the matrices  $A$  is arbitrary, from a mathematical point of view.

As we will show in the next paragraph, quadratic, linear, and constant terms in the algorithmic domain, eqn. (59), translate one-to-one into quadratic, linear and constant terms in the network domain. We therefore establish the notation  $\overline{Q}$ ,  $\overline{W}$ , and  $\overline{b}$  (with an overbar) to refer to 3rd-order tensors, matrices and biases in the neural domain, respectively. These are exactly the function  $\mathbf{Q}$  arguments that we had deemed feasible in eqn. (53) (there introduced without the overbar).

**Neural dynamics.** The neural dynamics for implementing each of the computations in eqn. (59) are as follows.

Quadratic terms:  $\partial_t z = \mathbf{yQx}$  is implemented via

$$\partial_t r^z = r^y \overline{Q} r^x \quad \text{with} \quad \overline{Q}_{ijk} \stackrel{\text{def}}{=} A_{i\alpha}^{z\dagger} Q_{\alpha\beta\gamma} A_{\beta j}^y A_{\gamma k}^x, \quad \text{in short:} \quad \overline{Q} \stackrel{\text{def}}{=} A^{z\dagger} (\mathbf{Q} A^y A^x). \quad (63)$$

Proof:

$$(\partial_t z)_i = (\partial_t A^z r^z)_i = (A^z \partial_t r^z)_i = A_{ij}^z (r^y \overline{Q} r^x)_j = A_{ij}^z \overline{Q}_{jkl} r_k^y r_l^x = A_{ij}^z A_{j\alpha}^{z\dagger} Q_{\alpha\beta\gamma} A_{\beta k}^y A_{\gamma l}^x r_k^y r_l^x = \delta_{i\alpha} Q_{\alpha\beta\gamma} y_\beta x_\gamma = (\mathbf{yQx})_i. \quad (64)$$

Linear terms:  $\partial_t z = \mathbf{Wx}$  is implemented via

$$\partial_t r^z = \overline{W} r^x \quad \text{with} \quad \overline{W} = A^{z\dagger} \mathbf{W} A^x. \quad (65)$$

Proof:

$$\partial_t z = A^z \partial_t r^z = A^z \overline{W} r^x = A^z A^{z\dagger} \mathbf{W} A^x r^x = \mathbf{W} x. \quad (66)$$

Constant terms:  $\partial_t z = \mathbf{b}$  is implemented via

$$\partial_t r^z = \overline{b} \quad \text{with} \quad \overline{b} = A^{z\dagger} \mathbf{b}. \quad (67)$$

Proof:

$$\partial_t z = A^z \partial_t r^z = A^z \overline{b} = A^z A^{z\dagger} \mathbf{b} = \mathbf{b}. \quad (68)$$

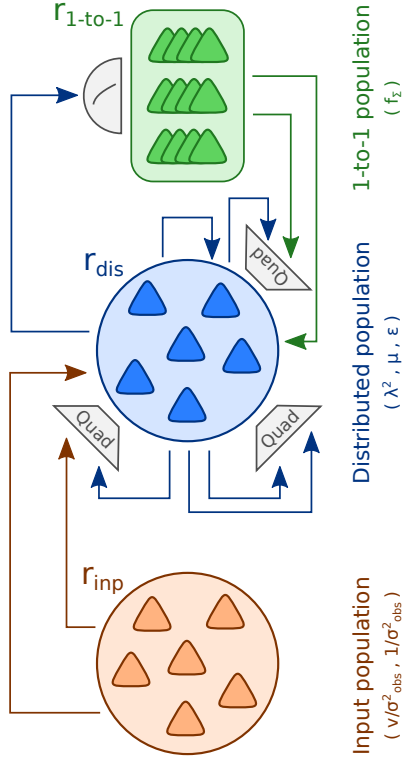
Linearity: Neural dynamics for linear combinations, e.g.,  $\partial_t z = \mathbf{W}^1 x + \mathbf{W}^2 y$  are simply the sum of the individual terms, e.g.,  $\partial_t r^z = \overline{W}^1 r^x + \overline{W}^2 r^y$ . The proof follows directly from the linearity of  $A^z$ .

Shared populations: Neural dynamics of variables encoded by the same population, e.g.,  $r = r^z = r^x$ , do not interfere. The proof follows from the orthogonality  $A^z A^{x\dagger} = A^x A^{z\dagger} = \mathbf{0}$  and from observing that every term in the above neural dynamics is led by an adjoint matrix  $A^{z\dagger}$  or  $A^{x\dagger}$ . Therefore, neural dynamics inducing changes in  $z$  do not convey any changes in  $x$ , and vice versa.

## 5.5 Recurrent network model for online motion structure inference

**Section 5.4** provides us with a straight-forward recipe for implementing eqn. (56)–(58) in a neural network. To keep the network as general as possible, the input variables,  $1/\sigma_{\text{obs}}^2$  and  $v/\sigma_{\text{obs}}^2$ , are encoded by an *input population*,  $r_{\text{inp}}$ . The latent variables,  $\lambda^2$ ,  $\mu$  and  $\epsilon$ , are encoded by a *distributed population*,  $r_{\text{dis}}$ . Both the input and distributed population employ a distributed code with arbitrary readout matrices  $A$  obeying the orthonormality conditions stated in **Section 5.4**. For the distributed population, the activation function,  $f_i$ , in eqn. (53) is simply the identity function. We refrain from restating the exact neural dynamics here because they are obtained directly by translating the terms in eqn. (56)–(58) into their neural counterparts by means of eqn. (63), (65), and (67).

For a functioning network model, however, two pieces are missing: the input code, and handling of the function  $f_\Sigma(\lambda_i^2)$  as was promised in **Section 5.3**. These two pieces are discussed next.



**Figure S8. Network model for motion structure inference.**

The network is composed of three neuronal populations. The input population,  $r_{\text{inp}}$ , encodes the input variables,  $1/\sigma_{\text{obs}}^2$  and  $v/\sigma_{\text{obs}}^2$ , as a distributed code. The distributed population,  $r_{\text{dis}}$ , encodes the latent variables,  $\lambda^2$ ,  $\mu$  and  $\epsilon$ , as a distributed code. The one-to-one population,  $r_{1\text{-to-1}}$ , encodes the latent posterior uncertainty,  $f_{\Sigma}(\lambda^2)$ , as a one-to-one code. All of these variables can be read out linearly from the network firing rate, at any time. Synaptic connections within and between populations mediate linear (indicated as arrows) and quadratic (indicated as “Quad” boxes) interactions. The non-linear function  $f_{\Sigma}$  is implemented by a leaky integrate-and-fire type response (indicated by the half-circle).

Overall, the network implements eqn. (56)–(58) of the algorithmic domain and, thereby, emulates the adiabatic observer model given by eqn. (43) + (44).

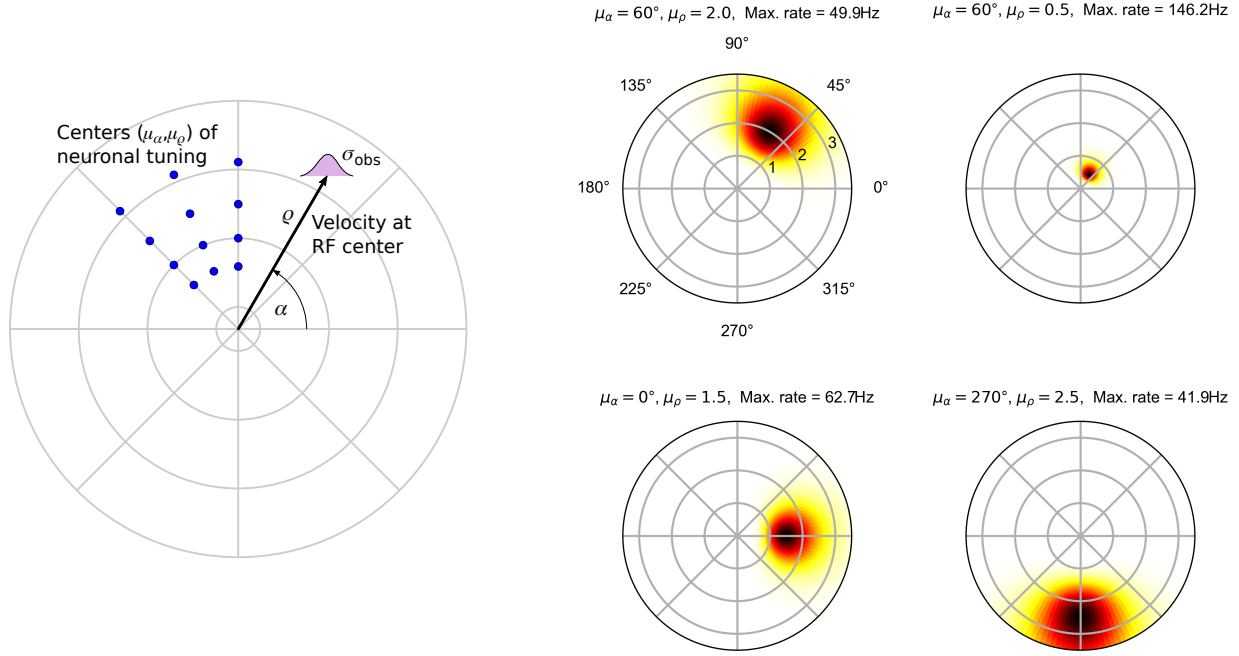
**Connecting the input.** The input is fed into the network externally and is thus by definition not controlled by internal dynamics of the network. Nonetheless, the activity  $r_{\text{inp}}(v/\sigma_{\text{obs}}^2, 1/\sigma_{\text{obs}}^2)$ , which is a function of the input variables, is required to support linearly decoding  $v/\sigma_{\text{obs}}^2$  and  $1/\sigma_{\text{obs}}^2$  via known readout matrices  $A^v$  and  $A^{\sigma}$ . Note that no adjoint matrices are required for the input. While any valid input code can be used in our generic network model, finding activation functions grounded in biological experiments together with matching readout matrices is typically non-trivial. We present one such input model, which resembles fundamental response properties of area MT, in **Section 5.6**.

**Handling  $f_{\Sigma}(\lambda_i^2)$ .** The non-linearity of the function  $f_{\Sigma}(\lambda_i^2)$ , given by eqn. (42), prohibits a direct incorporation of the effect of  $\lambda^2$  on its dependent variables in eqn. (56) and eqn. (57), within the computational framework of **Section 5.4**. The core reason is that the linear readout  $A^{\lambda} r_{\text{dis}}$  does not commute with the square-root function. Yet, it turns out that  $f_{\Sigma}(\lambda_i^2)$  can be incorporated into the network model quite easily owing to its simple functional form. Since  $f_{\Sigma}$  keeps all motion components separate,  $f_{\Sigma}(\lambda_m^2) = \text{const}_m \cdot (-1 + \sqrt{1 + \text{const}_m \lambda_m^2})$ , we can employ a dedicated population,  $r_{1\text{-to-1}}$ , using a one-to-one coding model, which assigns one neuron (or small sub-population) to each component of the posterior variance:

$$r_{1\text{-to-1},m} = \frac{1}{A_{mm}^{\Sigma}} f_{\Sigma}(A_{m*}^{\lambda} r_{\text{dis}}) . \quad (69)$$

Here we have used that  $\lambda_m^2 = A_{m*}^{\lambda} r_{\text{dis}}$  can be read out linearly from the population’s activity. The coefficient  $A_{mm}^{\Sigma}$  scales the firing rate of  $r_{1\text{-to-1},m}$ . This leads to a neurally plausible activation function of  $r_{1\text{-to-1}}$ : In **Fig. S7**, replace  $\lambda_m^2$  by the “input current”  $A_{m*}^{\lambda} r_{\text{dis}}$  on the x-axis, and  $f_{\Sigma}$  by  $r_{1\text{-to-1}}$  on the y-axis. This is reminiscent of the firing response of leaky integrate-and-fire neurons or, more generally, Type I neurons, as a function of the input current. Finally, we can read out  $f_{\Sigma}$  linearly from  $r_{1\text{-to-1}}$  via readout matrix  $A^{\Sigma} = \text{diag}[(A_{11}^{\Sigma}, \dots, A_{MM}^{\Sigma})]$ , thereby allowing us to apply the formalism of **Section 5.4** to  $f_{\Sigma}(\lambda^2)$  (which acts as a variable).

In the above argumentation, we have made two simplifying assumption. First, we have assumed an instantaneous response for  $r_{1\text{-to-1}}$  instead of the low-pass filtered response of eqn. (53). Since  $f_{\Sigma}$  varies only on the long time scale  $\tau_{\lambda}$ , eqn. (69) could easily be replaced by a low-pass filtered version with  $f_i := f_{\Sigma}/A_{mm}^{\Sigma}$  being the neurons’ activation function in eqn. (53). Secondly, we notice that, strictly,  $f_{\Sigma}$  depends not only on  $\lambda^2$ , but also on  $1/\sigma_{\text{obs}}^2$ . While the quadratic interaction between these variables, as expressed by eqn. (42), is covered by the theory, we decided to reduce the complexity of the network model by assuming a fixed default value for  $\sigma_{\text{obs}}^2$  in the computer simulations.



**Figure S9. Tuning functions of MT neurons.** *Left:* The tuning of MT neurons, in response to a stimulus with direction  $\alpha$ , speed  $\rho$  and observation noise  $\sigma_{\text{obs}}$ , is separable in polar coordinates. In a local population, the neurons' preferred velocity tuning covers directions,  $\mu_\alpha$ , uniformly, while the density of neurons tuned to speed,  $\mu_\rho$ , decreases for higher speed. Note that all quantities refer to a local coordinate system centered at the receptive field (RF) center. So, the coordinates here are not to be confused with the coordinate system in **Fig. S6** which describes RF locations. *Right:* Tuning function according to eqn. (70) for four example neurons. Tuning centers and maximum firing rates are given in the axes titles. Parameters (using Python indexing, i.e.,  $n_\rho = 0, \dots, N_\rho - 1$ ):  $\psi = 0.1$ ,  $\sigma_{\text{obs}}^2 = (0.05/3)^2$ ,  $N_\alpha = 16$ ,  $N_\rho = 12$ ,  $\mu_\rho(n_\rho) = \rho_{\text{min}} + d_\rho n_\rho^{1.25}$ ,  $d_\rho = (\rho_{\text{max}} - \rho_{\text{min}}) / (N_\rho - 1)^{1.25}$ ,  $\rho_{\text{min}} = 0.1$ ,  $\rho_{\text{max}} = 8.0$ ,  $\kappa_\alpha = 1/0.35^2$ ,  $\sigma_\rho^2 = 0.35^2$ .

Again, an extension respecting the explicit  $\sigma_{\text{obs}}^2$ -dependence would be straight-forward.

**The complete network model.** Plugging all of the components together, we obtain the network model shown in **Fig. S8**. This network emulates the adiabatic observer model given by eqn. (43) + (44).

## 5.6 Neural coding of the input: an example for area MT

While we aimed to leave the neural code for all *latent* variables as generic as possible in the network examples, we specify an *input* code that respects known response properties of area MT. In the following, we present the tuning functions for the input neurons which are derived from models and properties in the literature on area MT (Born and Bradley, 2005; Nover, Anderson, and DeAngelis, 2005; Kregelberg, 2006). Their most important computational property is that they support linear readout of  $v/\sigma_{\text{obs}}^2$  and  $1/\sigma_{\text{obs}}^2$  in Cartesian coordinates.

We will proceed in three steps. First, we define the tuning functions in polar coordinates since response properties are commonly presented in this coordinate system in the experimental literature. Second, we state some helpful mathematical properties of the proposed tuning functions. Third, we provide the readout matrices  $A^v$  and  $A^\sigma$  and demonstrate how they accurately decode the relevant variables,  $v/\sigma_{\text{obs}}^2$  and  $1/\sigma_{\text{obs}}^2$ .

**The tuning function in polar coordinates.** Commonly, MT tuning is characterized in polar coordinates because the tuning function becomes separable, i.e., it factorizes into a product of functions over different variables. Owing to the retinotopic organization of MT, we consider for the remainder of this subsection a local population of neurons sharing the same spatial receptive field. Those neurons are indexed, as shown in **Fig. S9 (left)**, by  $n_\alpha = 1..N_\alpha$  and  $n_\rho = 1..N_\rho$  according to their preferred direction,  $\mu_\alpha$ , and speed (absolute value of velocity),  $\mu_\rho$ , respectively. We use the following tuning function in response to a stimulus with direction  $\alpha \in [0, 2\pi)$ , speed  $\rho \geq 0$ , and observation



noise  $\sigma_{\text{obs}}^2$ :

$$f(\alpha, \rho, \sigma_{\text{obs}}^2; n_\alpha, n_\rho) = f_\sigma(\sigma_{\text{obs}}^2) \cdot f_\alpha(\alpha; n_\alpha) \cdot f_\rho(\rho; n_\rho) \quad (70)$$

$$\text{with} \quad f_\sigma(\sigma_{\text{obs}}^2) = \frac{\psi}{\sigma_{\text{obs}}^2}, \quad (71)$$

$$f_\alpha(\alpha; n_\alpha) = \frac{d_\alpha}{2\pi I_0(\kappa_\alpha)} e^{\kappa_\alpha \cos(\alpha - d_\alpha n_\alpha)}, \quad (72)$$

$$f_\rho(\rho; n_\rho) = \frac{\mu'_\rho(n_\rho)}{\sqrt{2\pi\sigma_\rho^2 \mu_\rho(n_\rho)}} e^{-\frac{(\log(\rho) - \log(\mu_\rho(n_\rho)))^2}{2\sigma_\rho^2}}. \quad (73)$$

Example tuning functions are shown in **Fig. S9 (right)**. Eqn. (70) is composed of sub-functions for the noise  $f_\sigma$ , motion direction  $f_\alpha$ , and motion speed  $f_\rho$ , which employ a range of parameters: The overall (maximum) firing rate is scaled by  $\psi$ . The angle between cells' preferred direction is  $d_\alpha = 2\pi/N_\alpha$ , such that neuron  $n_\alpha$ 's preferred direction is  $d_\alpha n_\alpha$ . The directional tuning width is described by  $\kappa_\alpha$  (formally,  $\kappa_\alpha$  is the precision parameter of a von-Mises-distribution density function, and  $I_0(\kappa_\alpha)$  is the modified Bessel function of order 0 for normalization). Neuron  $n_\rho$ 's preferred speed is given by function  $\mu_\rho(n_\rho)$ , with  $\mu'_\rho$  denoting the function's derivative. Finally, the width of speed tuning is controlled by  $\sigma_\rho^2$ . Let us briefly discuss how eqn. (70) captures known properties of MT:

- Neurons are tuned to speed (absolute value of velocity) and direction (almost entirely into only one direction, not the opposite direction).
- Direction tuning is commonly described by a von Mises density function. Preferred directions cover the circle roughly isotropically, here via  $d_\alpha n_\alpha$ .
- Speed tuning can be described by a log-normal function of the speed  $\rho$ . The density of speed tuning centers in MT has been reported to decrease for larger speeds, which can be captured by the function  $\mu_\rho(n_\rho)$ .
- Activity is modulated by contrast (via  $\sigma_{\text{obs}}^2$ ), with lower contrast (higher noise) attenuating the overall firing rate.

We make the simplifying assumptions that (i) all neurons have the same firing rate scaling factor  $\psi$ , and (ii) that the tuning widths, given by  $\kappa_\alpha$  and  $\sigma_\rho^2$ , are "not too wide". The meaning of "not too wide" will become clear in the following mathematical consideration.

**Mathematical properties of the tuning function.** We next discuss some useful properties of the components of the above tuning function. First, we note that  $f_\alpha$  and  $f_\rho$  have the form of known probability density functions over the neuron indices  $n_\alpha$  and  $n_\rho$ , respectively. In particular, they integrate to one in the limit of many, narrowly spaced neurons:

$$\int_0^{2\pi/d_\alpha} f_\alpha(\alpha; n_\alpha) dn_\alpha = \int_0^{2\pi} e^{\kappa_\alpha \cos(\alpha - d_\alpha n_\alpha)} / (2\pi I_0(\kappa_\alpha)) d(d_\alpha n_\alpha) = 1 \quad (74)$$

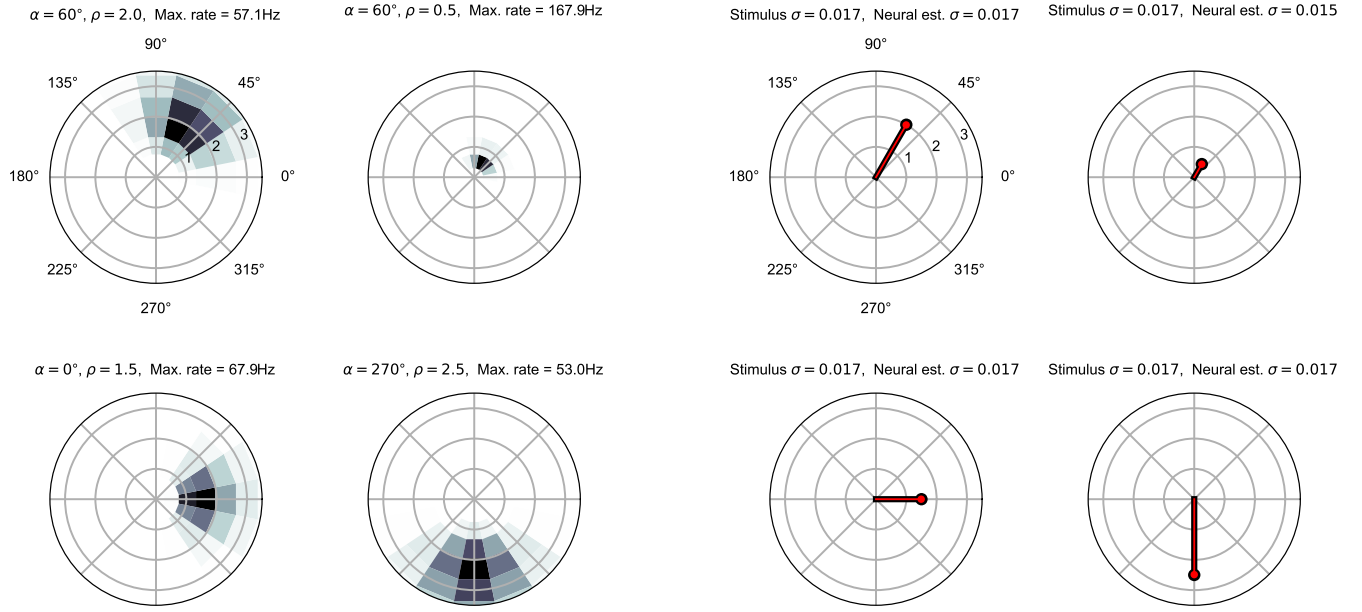
$$\text{and} \quad \int_0^\infty f_\rho(\rho; n_\rho) dn_\rho = \int_0^\infty \frac{1}{\sqrt{2\pi\sigma_\rho^2 \mu_\rho}} e^{-\frac{(\log(\rho) - \log(\mu_\rho))^2}{2\sigma_\rho^2}} d\mu_\rho = 1. \quad (75)$$

Furthermore, the distributional forms give rise to nice moments w.r.t. the tuning centers  $d_\alpha n_\alpha$  and  $\mu_\rho(n_\rho)$ :

$$\langle e^{i d_\alpha n_\alpha} \rangle_{f_\alpha} = \frac{I_1(\kappa_\alpha)}{I_0(\kappa_\alpha)} e^{i\alpha} \stackrel{\text{large } \kappa_\alpha}{\approx} e^{i\alpha} = \begin{pmatrix} \cos \alpha \\ \sin \alpha \end{pmatrix} \quad (76)$$

$$\text{and} \quad \langle \mu_\rho(n_\rho) \rangle_{f_\rho} = \rho e^{\sigma_\rho^2/2} \stackrel{\text{small } \sigma_\rho}{\approx} \rho. \quad (77)$$

We now understand how narrow (i.e., "not too wide") tuning functions enable reading out the encoded direction,  $\alpha$ , and speed,  $\rho$ : large  $\kappa_\alpha$  and small  $\sigma_\rho^2$  afford the approximations in eqn. (76) and (77). Further, the mathematical relations highlight that  $\kappa_\alpha$  and  $\sigma_\rho^2$  could be modulated by the observation noise  $\sigma_{\text{obs}}^2$  without changing the ability to encode/decode the input.



**Figure S10. Linear readout of input statistics from population responses.** *Left:* Population response when encoding four example stimuli. The population consists of 192 neurons with the parameters given in **Fig. S9**. For clarity, only neurons with  $\mu_\rho < 3.5$  are shown. *Right:* Linear readout of  $1/\sigma_{\text{obs}}^2$ ,  $v_x/\sigma_{\text{obs}}^2$  and  $v_y/\sigma_{\text{obs}}^2$  from the population activities on the left via weights given by eqn. (78). Shown are, in polar coordinates, the stimulus ground truth (black) and the estimate by the linear readout (red). The estimated uncertainty is provided in the axes titles.

**Linear readout of input statistics.** With the above mathematical properties at hand, we identify matrices  $A^\sigma$  and  $A^v$  for linear readout:

$$A_{k,(n_\alpha, n_\rho)}^\sigma = \frac{1}{\psi} \quad \text{and} \quad A_{k,(n_\alpha, n_\rho)}^v = \frac{1}{\psi} \begin{pmatrix} \cos d_\alpha n_\alpha \\ \sin d_\alpha n_\alpha \end{pmatrix} \mu_\rho(n_\rho), \quad (78)$$

because reading out with these matrices from the MT-population of the  $k$ -th observable yields:

$$\int \int A_{k,(n_\alpha, n_\rho)}^\sigma f(\alpha_k, \rho_k, \sigma_{\text{obs},k}^2; n_\alpha, n_\rho) dn_\alpha dn_\rho = \frac{1}{\sigma_{\text{obs},k}^2} \quad (79)$$

$$\int \int A_{k,(n_\alpha, n_\rho)}^v f(\alpha_k, \rho_k, \sigma_{\text{obs},k}^2; n_\alpha, n_\rho) dn_\alpha dn_\rho = \frac{\rho_k}{\sigma_{\text{obs},k}^2} \begin{pmatrix} \cos \alpha_k \\ \sin \alpha_k \end{pmatrix} = \begin{pmatrix} v_x/\sigma_{\text{obs},k}^2 \\ v_y/\sigma_{\text{obs},k}^2 \end{pmatrix}. \quad (80)$$

Examples of the population response to four motion stimuli is shown in **Fig. S10 (left)** for a population of 192 neurons. The tuning centers span 12 radii (“speed”) and 16 angles (“direction”). In **Fig. S10 (right)**, the resulting linear readout (red) is shown alongside the ground truth (black). Even the coarse coverage of the stimulus space by 192 neurons is sufficient for a faithful reconstruction of the stimulus. As an interesting observation, while neurons tuned to higher speeds have wider tuning curves, this does not imply that their activity would encode heightened uncertainty:  $\sigma_{\text{obs}}$  is identical in all of the examples in **Fig. S9** and **Fig. S10**.

In summary, we have identified with eqn. (70) an MT tuning function that supports linear readout of the variables  $v/\sigma_{\text{obs}}^2$  and  $1/\sigma_{\text{obs}}^2$  via matrices  $A^v$  and  $A^\sigma$ .

## 6. Captions to the supplemental videos

**Video S1: Johansson experiment.** *Top:* 3-dot motion display. *Bottom left:* Evolution of motion strengths,  $\lambda_m(t)$ . *Bottom right:* Motion sources,  $s_m(t)$ . The stimulus is decomposed into horizontal shared motion (magenta) and vertical individual motion (green) for the central dot.

**Video S2: Duncker wheel.** *Top:* Duncker wheel motion display. *Bottom left:* Evolution of motion strengths,  $\lambda_m(t)$ . *Bottom right:* Motion sources,  $s_m(t)$ . The stimulus is decomposed into rightward shared motion (magenta) and clockwise rotation (green) for the dot on the rim.

**Video S3: Motion illusion from Lorenceau (1996).** Motion displays without and with motion noise are presented. Multiple noise levels are used because the noise threshold for inducing the perceptual switch is participant-dependent. To prevent priming on a certain grouping, different random groups were drawn for every noise level, in this demo video. (The simulation in Fig. 4 used, apart from  $\sigma_{\text{obs}}^2$ , identical inputs.)

**Video S4: Location-indexed stimulus with varying fraction of shared motion.** Every 10 s, the fraction of shared motion across the apertures increases by  $1/8$ , ranging from  $q=1/8$  (almost independent motion) to  $q=7/8$  (almost fully correlated motion). Steps in  $q$  are marked by brief flashes of gray squares.

## References

- Beck, Jeffrey M, Peter E Latham, and Alexandre Pouget (2011). "Marginalization in neural circuits with divisive normalization". In: *Journal of Neuroscience*.
- Bill, Johannes et al. (2020). "Hierarchical structure is employed by humans during visual motion perception". In: *Proceedings of the National Academy of Sciences*.
- Bishop, Christopher M (2006). *Pattern recognition and machine learning*. springer.
- Born, Richard T. and David C. Bradley (July 2005). "STRUCTURE AND FUNCTION OF VISUAL AREA MT". In: *Annual Review of Neuroscience*.
- Dayan, Peter and Laurence F Abbott (2001). *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series.
- Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)*.
- Drugowitsch, Jan et al. (2012). "The cost of accumulating evidence in perceptual decision making". In: *Journal of Neuroscience*.
- Gardiner, Crispin (2009). *Stochastic methods*. Springer Berlin.
- Jazwinski, Andrew H (2007). *Stochastic processes and filtering theory*. Courier Corporation.
- Kalman, Rudolph E (1960). "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering*.
- Kalman, Rudolph E and Richard S Bucy (1961). "New results in linear filtering and prediction theory". In: *Journal of Basic Engineering*.
- Krekelberg, B. (Aug. 2006). "Interactions between Speed and Contrast Tuning in the Middle Temporal Area: Implications for the Neural Code for Speed". In: *Journal of Neuroscience*.
- Manning, Catherine, Rory Trevelyan Thomas, and Oliver Braddick (2018). "Can speed be judged independent of direction?" In: *Journal of vision*.
- Moscattelli, Alessandro et al. (2019). "Motion direction, luminance contrast, and speed perception: an unexpected meeting". In: *Journal of vision*.
- Nover, Harris, Charles H. Anderson, and Gregory C. DeAngelis (Oct. 2005). "A Logarithmic, Scale-Invariant Representation of Speed in Macaque Middle Temporal Area Accounts for Speed Discrimination Performance". In: *Journal of Neuroscience*. Publisher: Society for Neuroscience Section: Behavioral/Systems/Cognitive.
- Salinas, Emilio and Laurence F Abbott (1996). "A model of multiplicative neural responses in parietal cortex". In: *Proceedings of the national academy of sciences*.
- Weiss, Yair, Eero P Simoncelli, and Edward H Adelson (2002). "Motion illusions as optimal percepts". In: *Nature neuroscience*.