EXPERIMENTAL METHODS

Animal care and behavioral experiments

Unless otherwise noted, behavioral recordings were performed on 8–16-week-old C57/BL6 mice (The Jackson Laboratory stock no. 000664). Mice were transferred to our colony at 6-8 weeks of age and housed in a reverse 12-hour light/12-hour dark cycle. We single-housed mice after stereotactic surgery, and group-housed them otherwise. On recording days, mice were brought to the laboratory, habituated in darkness for at least 20 minutes, and then placed in an open field arena for 30-60 mins. We recorded 6 male mice for 10 sessions (6 hours) in the initial round of open field recordings; and 5 male mice for 52 sessions (50 hours) during the accelerometry recordings. The dopamine photometry recordings were obtained from a recent study[1]. They include 6 C57/BL6 mice and 8 DAT-IRES-cre (The Jackson Laboratory stock no. 006660) mice of both sexes, recorded for 378 sessions. Of these, we selected a random subset of 95 sessions (~50 hours) for benchmarking keypoint-MoSeq.

Stereotactic surgery procedures

For all stereotactic surgeries, mice were anaesthetized using 1–2% isoflurane in oxygen, at a flow rate of 1 L/min for the duration of the procedure. Anterior-posterior (AP) and medial-lateral (ML) coordinates were zeroed relative to bregma, the dorso-ventral (DV) coordinate was zeroed relative to the pial surface, and coordinates are in units of mm. For dopamine recordings, 400nL of AAV5.CAG.dLight1.1 (Addgene #111067, titer: 4.85×10^{12}) was injected at a 1:2 dilution into the DLS (AP 0.260; ML 2.550; DV −2.40) and a single 200-μm diameter, 0.37–0.57 NA fiber cannula was implanted 200 μm above the injection site (see ref[1] for additional details). For accelerometry recordings, we surgically attached a millmax connector (DigiKey ED8450-ND) and head bar to the skull and secured it with dental cement (Metabond). A 9 degree-of-freedom absolute orientation inertial measurement unit (IMU; Bosch BN0055) was mounted on the millmax connector using a custom printed circuit board (PCB) with a net weight below 1g.

Data acquisition from the IMU

The IMU was connected to a Teensy microcontroller, which was programmed using the Adafruit BNO055 library with default settings (sample rate: 100 Hz, units: m/s^2). To synchronize the IMU measurements and video recordings, we used an array of near infrared LEDs to display a rapid sequence of random 4-bit codes that updated throughout the recording. The code sequence was later extracted from the behavioral videos and used to fit a piecewise linear model between timestamps from the videos and timestamps from the IMU.

Recording setup

For the initial set of open field recordings (Fig 1-3, 4a-g Fig 6g-l), mice were recorded in a square arena with transparent floor and walls (30cm length and width). Microsoft Azure Kinect cameras captured simultaneous depth and near-infrared video at 30Hz. Six cameras were used in total: one above, one below, and four side cameras at right angles at the same height as the mouse. For the accelerometry recordings, we used a single Microsoft Azure Kinect camera placed above the mouse, and an arena with transparent floor and opaque circular walls (45cm diameter). Data was transferred from the IMU using a light-weight tether attached to a custom-built active commutator. For the dopamine perturbation experiments, we used a slightly older camera model – the Microsoft Kinect 2 – to capture simultaneous depth and near-infrared at 30Hz. The recording arena was circular with opaque floor and walls (45cm diameter). Photometry signals were conveyed from the mouse using a fiber-optic patch cord attached to a passive commutator.

COMPUTATIONAL METHODS

Processing depth videos

Applying MoSeq to depth videos involves: (1) mouse tracking and background subtraction; (2) egocentric alignment and cropping; (3) principal component analysis (PCA); (4) probabilistic modeling. We applied steps (2-4) as described in the MoSeq2 pipeline[2]. For step (1), we trained a convolutional neural network (CNN) with a Unet++[3] architecture to segment mouse from background using ~5000 hand-labeled frames as training data.

Keypoint tracking

We used CNNs with an HRNet[4] architecture (https://github.com/stefanopini/simple-HRNet) with a final stride of 2 for pose tracking. The networks were trained on ~1000 hand-labeled frames each for the overhead, below-floor, and side-view camera angles. Frame-labelling was crowdsourced through a commercial service (Scale AI). For the overhead camera, we tracked two ears and 6 points along the dorsal midline (tail base, lumbar spine, thoracic spine, cervical spine, head, and nose). For the below-floor camera, we tracked the tip of each forepaw, the tip and base of each hind paw, and four points along the ventral midline (tail base, genitals, abdomen, and nose). For the side cameras, we tracked the same eight points as for the overhead camera, and also included the six limb points that were used for the below-floor camera (14 total). We trained a separate CNN for each camera angle. Target activations were formed by centering a Gaussian with 10px standard deviation on each keypoint. We used the location of the maximum pixel in each output channel of the neural network to determine keypoint coordinates and used the value at that pixel to set the confidence score. The resulting mean absolute error (MEA) between network detections and manual annotations was 2.9 pixels (px) for the training data and 3.2 px for heldout data. We also trained DeepLabCut and SLEAP models on the overhead-camera and below-floor-camera datasets. For DeepLabCut, we used version 2.2.1, setting the architecture to resnet50 architecture and the “pos_dist_thresh” parameter to 10, resulting in train and test MEAs of 3.4 px and 3.8 px respectively. For SLEAP, we used version 1.2.3 with the baseline_large_rf.single.json configuration, resulting in train and test MEAs of 3.5 px and 4.7 px.

3D pose inference

Using 2D keypoint detections from six cameras, 3D keypoint coordinates were triangulated and then refined using GIMBAL, a model-based approach that leverages anatomical constraints and motion continuity[5]. To fit GIMBAL, we computed initial 3D keypoint estimates using robust triangulation (i.e. by taking the median across all camera pairs, as in 3D-DeepLabCut[6]) and then filtered to remove outliers using the EllipticEnvelope method from sklearn; We then fit the skeletal parameters and directional priors for GIMBAL using expectation maximization with 50 pose states (see ref[5] for details). Finally, we applied the fitted GIMBAL model to each recording, using the following parameters for all keypoints: obs_outlier_variance=1e6, obs_inlier_variance=10, pos_dt_variance=10. The latter parameters were chosen based on the accuracy of the resulting 3D keypoint estimates, as assessed from visual inspection.

Inferring model-free changepoints

We defined changepoints as sudden, simultaneous shifts in the trajectories of multiple keypoints. We detected them using a procedure similar to the filtered derivative algorithm described in ref[7], but with changes to emphasize simultaneity across multiple keypoints. The changes account for the lower dimensionality of keypoint data compared to depth videos, and for the unique noise structure of markerless keypoint tracking, in which individual keypoints occasionally jump a relatively large distance due to detection errors. Briefly, the new procedure first defines a continuous change score by: (1)

calculating the rate of each in each keypoint coordinate; (2) quantifying simultaneity in the change-rates across keypoints; (3) transforming the signal based on statistical significance with respect to a temporally shuffled null distribution; (4) identifying local peaks in the resulting significance score. The details of each step are as follows.

1) **Calculating rates of change:** We transformed the keypoint coordinates on each frame by centering and aligned them along the tail-nose axis. We then computed the derivative of each coordinate for each keypoint, using a sliding window of length 3 as shown below, where x_t denotes the value of a coordinate at time t.

$$\dot{x}_t \approx \frac{1}{3}(x_{t+3} + x_{t+2} + x_{t+1} - x_{t-1} - x_{t-2} - x_{t-3})$$

2) **Quantifying simultaneous changes:** The derivatives for each keypoint were Z-scored and then binarized with a threshold. We then counted the number of threshold crossings on each frame and smoothed the resulting time-series of counts using a Gaussian filter with a one-frame kernel. The value of the threshold was chosen to maximize the total number of detected changepoints.

3) **Comparing to a null distribution:** We repeated step (2) for 1000 shuffled datasets, in which each keypoint trajectory was cyclically permuted by a random interval. Using the shuffles as a null distribution, we computed a P-value for each frame and defined the final change score as $-\log_{10}(\text{pval})$

4) **Identifying local peaks in the change score:** We identified local peaks in the change score s_t, i.e., times t for which $s_{t-1} < s_t > s_{t+1}$. Peaks were classified as statistically significant when they corresponded to a p-value below 0.01, which was chosen to control the false-discovery rate at 10%. The statistically significant peaks were reported as changepoints for downstream analysis.

Spectral Analysis

To analyze keypoint jitter, we quantified the magnitude of fluctuations across a range of frequencies by computing a spectrogram for each keypoint along each coordinate axis. Spectrograms were computed using the python function scipy.signal.spectrogram with nperseg=128 and noverlap=124. The spectrograms were then combined through averaging: each keypoint was assigned a spectrogram by averaging over the two coordinate axes, and the entire animal was assigned a spectrogram by averaging over all keypoints.

We used the keypoint-specific spectrograms to calculate cross-correlations with $-\log_{10}(\text{neural network detection confidence})$, as well as the "error magnitude" (Fig 2f). Error magnitude was defined as the distance between the detected 2D location of a keypoint (based on a single camera angle) and a reprojection of its 3D position (based on consensus across six camera angles; see "3D pose inference" above). We also computed the cross-correlation between nose- and tail-base-fluctuations at each frequency, as measured by the overhead and below-floor cameras respectively. Finally, we averaged spectral power across keypoints to compute the cross-correlation with model transition rates (Fig 2f), defined as the per-frame probability of a state transitions across 20 model restarts.

Applying keypoint-MoSeq

The initial open field recordings (Fig 1-4), as well as the accelerometry, dopamine, and two benchmark datasets were modeled separately. Twenty models with different random seeds were fit for each dataset (except for the accelerometry data, in which case one model was fit).

Modeling consisted of two phases: (1) Fitting an autoregressive hidden Markov model (AR-HMM) to a fixed pose trajectory derived from PCA of egocentric-aligned keypoints; (2) Fitting a full keypoint-MoSeq model initialized from the AR-HMM. References in the text to "MoSeq applied to keypoints" or "MoSeq (keypoints)", e.g., in Figs 2-3, refer to output of step (1). Both steps are described below, followed by a detailed description of the model and inference algorithm in the mathematical modeling section. In all cases, we excluded rare states (frequency < 0.5%) from downstream analysis. We have made the code available as a user-friendly package, available at Moseq4all.org.

1) Fitting an initial AR-HMM*:*

We first modified the keypoint coordinates, defining keypoints with confidence below 0.5 as missing data and in imputing their values via linear interpolation, and then augmenting all coordinates with a small amount of random noise; The noise values were uniformly sampled from the interval [-0.1, 0.1] and helped prevent degeneracy during model fitting. Importantly, these preprocessing steps were only applied during AR-HMM fitting – the original coordinates were used when fitting the full keypoint-MoSeq model.

Next, we centered the coordinates on each frame, aligned them using the tail-nose angle, and then transformed them using PCA with whitening. The number of principal components (PCs) was chosen for each dataset as the minimum required to explain 90% of total variance. This resulted in 4 PCs for the overhead camera 2D datasets, 6 PCs for the below-floor-camera 2D datasets, and 6 PCs for the 3D dataset.

We then used Gibbs sampling to infer the states and parameters of an AR-HMM, including the state sequence z, the autoregressive parameters A, b, Q, and the transition parameters π, β. The hyper-parameters for this step, listed in the mathematical modeling section below, were generally identical to those in the original depth-MoSeq model[7]. The one exception was κ which we adjusted separately for each dataset to ensure a median state duration of 400ms.

2) Fitting a full keypoint-MoSeq model:

We next fit the full set of variables for keypoint-MoSeq, which include the AR-HMM variables mentioned above, as well as the location v and heading h, latent pose trajectory x, per-keypoint noise level σ^2, and per-frame/per-keypoint noise scale s. Fitting was performed using Gibbs sampling for 500 iterations, at which point the log joint probability appeared to have stabilized.

The hyper-parameters for this step are enumerated in the mathematical modeling section below. In general, we used the same hyper-parameter values across datasets. The two exceptions were κ, which again had to be adjusted to maintain a median state duration of 400ms, and s_0, which determines a prior on the noise scale. Since low-confidence keypoint detections often have high error, we set s_0 using a logistic curve that transitions between a high-noise regime ($s_0 = 100$) for detections with low confidence and a low-noise regime ($s_0 = 1$) for detections with high confidence:

$$s_0 = 1 + 100\left(1 + e^{20(\text{confidence}-0.4)}\right)^{-1}$$

Trajectory plots

To visualize the modal trajectory associated with each syllable (Fig 3e), we (1) computed the full set of trajectories for all instances of all syllables (2) used a local density criterion to identify a single representative instance of each syllable (3) computed a final trajectory using the nearest neighbors of the representative trajectory.

1) Computing the trajectory of individual syllable instances: Let y_t, v_t, and h_t denote the keypoint coordinates, centroid and heading of the mouse at time t, and let $F(v, h; y)$ denote the rigid transformation that egocentrically aligns y using centroid v and heading h. Given a syllable instance with onset time T, we

computed the corresponding trajectory X_T by centering and aligning the sequence of poses $(y_{T-5}, \ldots, y_{T+15})$ using the centroid and heading on time T. In other words,

$$X_T = [F(v_T, h_T; y_{T-5}), \ldots, F(v_T, h_T; y_{T+15})]$$

2) Identifying a representative instance of each syllable: The collection of trajectories computed above can be thought of as a set of points in a high dimensional trajectory space (for K keypoints in 2D, this space would have dimension $40K$). Each point has a syllable label, and the segregation of these labels in the trajectory space represents the kinematic differences between syllables. To capture these differences, we computed a local probability density function for each syllable, and a global density function across all syllables. We then selected a representative trajectory X for each syllable by maximizing the ratio:

$$\frac{\text{local density}(X)}{\text{global density}(X)}$$

The density functions were computed as the mean distance from each point to its 50 nearest neighbors. For the global density, the nearest neighbors were selected from among all instances of all syllables. For the local densities, the nearest neighbors were selected from among instances of the target syllable.

3) Computing final trajectories for each syllable: For each syllable and its representative trajectory X, we identified the 50 nearest neighbors of X from among other instanes of the same syllable and then computed a final trajectory as the mean across these nearest neighbors. The trajectory plots in Fig 3e consist of 10 evenly-space poses along this trajectory, i.e., the poses at times $T-5, T-3, \ldots, T+13$.

Cross-syllable likelihoods

We defined each cross-syllable likelihood[7] as the probability (on average) that instances of one syllable could have arisen based on the dynamics of another syllable. The probabilities were computed based on the discrete latent states z_t, continuous latent states x_t, and autoregressive parameters A, b, Q output by keypoint-MoSeq. The instances $I(n)$ of syllable n were defined as the set of all sequences $(t_s, \ldots, t_e)$ of consecutive timepoints such that $z_t = n$ for all $t_s \leq t \leq t_e$ and $z_{t_s-1} \neq n \neq z_{t_e+1}$. For each such instance, one can calculate the probability $P\left(x_{t_s}, \ldots, x_{t_e} \middle| A_m, b_m, Q_m\right)$ that the corresponding sequence of latent states arose from the autoregressive dynamics of syllable m. The cross-syllable likelihood C_{nm} is defined in terms of these probabilities as

$$C_{nm} = \frac{1}{|I(n)|} \sum_{(t_s,\dots,t_e) \in \mathrm{I(n)}} \frac{(x_{t_s}, \dots, x_{t_e} | A_m, b_m, Q_m)}{(x_{t_s}, \dots, x_{t_e} | A_n, b_n, Q_n)}$$

Generating synthetic keypoint data

To generate the synthetic keypoint trajectories used for Extended Data Fig 3c, we fit a linear dynamical system (LDS) to egocentrically aligned keypoint trajectories and then sampled randomly generated outputs from the fitted model. The LDS was identical to the model underlying keypoint-MoSeq (see mathematical modeling section below), except that it only had one discrete state, lacked centroid ad heading variables, and allowed separate noise terms for the x- and y- coordinates of each keypoint.

Applying B-SOiD

B-SOiD is an automated pipeline for behavioral clustering that: (1) preprocesses keypoint trajectories to generate pose and movement features; (2) performs dimensionality reduction on a subset of frames using UMAP; (3) clusters points in the UMAP space; (4) uses a classifier to extend the clustering to all frames[8]. We fit B-SOiD separately for each dataset. In each case, steps 2-4 were performed 20 times with different random seeds, and the pipeline was applied with standard parameters; 50,000 randomly sampled frames were used for dimensionality reduction and clustering, and the min_cluster_size range was set to 0.5% - 1%. Since B-SOiD uses a hardcoded window of 100ms to calculate pose and movement features, we re-ran the pipeline with falsely inflated framerates for the window-size scan in Extended Data Fig 4a. In all analyses involving B-SOiD, rare states (frequency < 0.5%) were excluded from analysis.

Applying VAME

VAME is a pipeline for behavioral clustering that: (1) preprocesses keypoint trajectories and transforms them into egocentric coordinates; (2) fits a recurrent neural network (RNN); (3) clusters the latent code of the RNN[9]. We applied these steps separately to each dataset, in each case running step (3) 20 times with different random seeds. For step (1), we used the same parameters as in keypoint-MoSeq – egocentric alignment was performed along the tail-nose axis, and we set the pose_confidence threshold to 0.5. For step (2), we set time_window=30 and zdims=20 for all datasets, except for the zdim-scan in Extended Data Fig 4a. VAME provides two different options for step (3): fitting an HMM (default) or applying K-Means (alternative). We fit an HMM for all datasets and additionally applied K-Means to the initial open dataset. In general, we approximately matched the number of states/clusters in VAME to the number identified by keypoint-MoSeq, except when scanning over state number in Extended Data Fig 4a. In all analyses involving VAME, rare states (frequency < 0.5%) were excluded from analysis.

Applying MotionMapper

MotionMapper performs unsupervised behavioral segmentation by: (1) applying a wavelet transform to preprocessed pose data; (2) nonlinearly embedding the transformed data in 2D; (3) clustering the 2D data with a watershed transform[10]. We applied MotionMapper separately to each dataset using the python package https://github.com/bermanlabemory/motionmapperpy. In general, the data were egocentrically aligned along the tail-nose axis and then projected into 8 dimensions using PCA. 10 log-spaced frequencies between 0.25 and 15Hz were used for the wavelet transform, and dimensionality reduction was performed using tSNE. The threshold for watershedding was chosen to produce at least 25 clusters, consistent with keypoint-MoSeq for the overhead camera data. Rare states (frequency < 0.5%) were excluded from analysis. For the parameter scan in Extended Data Fig 4a, we varied the each of these parameters while holding the others fixed, including the threshold for watershedding, the number of initial PCA dimensions, and the frequency range of wavelet analysis. We also repeated a subset of these analyses using an alternative autoencoder-based dimensionality reduction approach, as described in the motionmapperpy tutorial (motionmapperpy/demo/motionmapperpy_mouse_demo.ipynb).

Predicting kinematics from state sequences

We trained decoding models based on spline regression to predict kinematic parameters (height, velocity, turn speed) from state sequences output by keypoint-MoSeq and other behavior segmentation methods (Fig 4e, Extended Data Fig 4c). Let z_t represent an unsupervised behavioral state sequence and let B denote a spline basis, where $B_{t,i}$ is the value of spline i and frame t. We generated such a basis using the “bs” function from the python package “patsy”, passing in five log-spaced knot locations (1.0, 2.0, 3.9, 7.7, 15.2, 30.0) and obtaining basis values over a 300-frame interval. This resulted in a 300-by-5 basis matrix B. The spline basis and state sequence were combined to form a $5N$-dimensional design matrix, where N is the number of distinct behavioral states. Specifically, for each instance $(t_s, \ldots, t_e)$ of state n (see “Cross-syllable likelihoods” section above for a definition of state instances), we inserted the first $t_e - t_s$ frames of B into dimensions $5n, \ldots, 5n+5$ of the design matrix, aligning the first frame of B to frame t_s in the design matix. Kinematic features were regressed against the design matrix using Ridge regression from scikit-learn and 5-fold cross-validation. We used a range of values from 10^{-3} to 10^{3} for the regularization parameter α and reported the results with greatest accuracy.

Rearing analysis

To compare the dynamics of rear-associated states across methods, we systematically identified all instances of rearing in our initial open field dataset. During a stereotypical rear, mice briefly stood on their hindlegs and extended their head upwards, leading to a transient increase in height from its modal value of 3cm-5cm to a peak of 7cm-10cm. Rears were typically brief, with mice exiting and then returning to a prone position within a few seconds. We encoded these features using the following criteria. First, rear onsets were defined as increases in height from below 5cm to above 7cm that occurred within the span of a second, with onset formally defined as the first frame where the height exceeded 5cm. Next, rear offsets were defined as decreases in height from above 7cm to below 5cm that occurred within the span of a second, with offset formally defined as the first frame where the height fell below 7cm. Finally, we defined complete rears as onset-offset pairs defining an interval with length between 0.5 and 2 seconds. Height was determined from the distribution of depth values in cropped, aligned and background-segmented videos. Specifically, we used the 98th percentile of the distribution in each frame.

Accelerometry processing

From the IMU we obtained absolute rotations r_y, r_p, r_r (yaw, pitch, and roll) and accelerations a_x, a_y, a_z (dorsal/ventral, posterior/anterior, left/right). To control for subtle variations in implant geometry and chip calibration, we centered the distribution of sensor readings for each variable on each session. We defined total acceleration as the norm of the 3 acceleration components:

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Similarly, we defined total angular velocity as the norm $|\omega|$ of rotation derivative:

$$\omega = \left(\frac{dr_y}{dt}, \frac{dr_p}{dt}, \frac{dr_r}{dt}\right)$$

Finally, to calculate jerk, we smoothed the acceleration signal with a 50ms Gaussian kernel, generating a time-series $\tilde{a}$, and then computed the norm of its derivative:

$$\text{jerk} = \left|\frac{d\tilde{a}}{dt}\right|$$

Aligning dopamine fluctuations to behavior states

For a detailed description of photometry data acquisition and preprocessing, see ref[1]. Briefly, photometry signals were: (1) ΔF/F0-normalized using a 5-second window; (2) adjusted against a reference to remove motion artefacts and other non-ligand-associated fluctuations; (3) z-scored using a 20-second sliding window; (4) temporally aligned to the 30Hz behavioral videos.

Given a set of state onsets (either for a single state or across all states), we computed the onset-aligned dopamine trace by averaging the dopamine signal across onset-centered windows. From the resulting traces, each of which can be denoted as a time-series of dopamine signal values $(d_{-T}, \ldots, d_T)$ we defined the total fluctuation size (Fig 5d) and temporal asymmetry (Fig 5e) as

$$\text{temporal asymmetry} = \frac{1}{15}\sum_{t=0}^{15} d_t - \frac{1}{15}\sum_{t=-15}^{0} d_t\,, \qquad \text{AUC} = \sum_{t=-15}^{15} |d_t|$$

A third metric – the average dopamine during each state (Extended Data Figure 6b) – was defined simply as the mean of the dopamine signal across all frames bearing that state label. For each metric, shuffle distributions were generated by repeating the calculation with a temporally reversed copy of the dopamine times-series.

Supervised behavior benchmark

Videos and behavioral annotations for the supervised open field behavior benchmark (Fig 4a-c) were obtained from (Bohnslav, 2019)[11]. The dataset contains 20 videos that are each 10-20 minutes long. Each video includes frame-by-frame annotations of five possible behaviors: locomote, rear, face groom, body groom, and defecate. We excluded “defecate” from the analysis because it was extremely rate (< 0.1% of frames).

For pose tracking we used DLC's SuperAnimal inference API that performs inference on videos without the need to annotate poses in those videos. Specifically, we used SuperAnimal-TopViewMouse that applies DLCRNet-50 as the pose estimation model11. Keypoint detections were obtained using DeepLabCut's API function deeplabcut.video_inference_superanimal. The API function uses a pretrained model called SuperAnimal-TopViewMouse and performs video adaptation that applies multi-resolution ensemble (i.e., the image height resized to 400, 500, 600 with a fixed aspect ratio) and rapid self-training (model trained on zero shot predictions with confidence above 0.1) for 1000 iterations to counter domain shift and reduce jittering predictions. The code to reproduce this analysis is:

```
videos = ['path_to_video']
superanimal_name = 'superanimal_topviewmouse'
scale_list = [400, 500, 600]

deeplabcut.video_inference_superanimal(videos,
  superanimal_name,
  videotype=".mp4",
  video_adapt = True,
  scale_list = scale_list)
```

Keypoint coordinates and behavioral annotations for the supervised social behavior benchmark (Fig 4d-f) were obtained from the CalMS21 dataset[12] (task1). The dataset contains 70 videos of resident-intruder interactions with frame-by-frame annotations of four possible behaviors: attack, investigate, mount, or other. All unsupervised behavior segmentation methods were fit to 2D keypoint data for the resident mouse.

We used four metrics[9] to compare supervised annotations and unsupervised states from each method. These included normalized mutual information, homogeneity, adjusted rand score, and purity. All metrics besides purity were computed using the python library scikit-learn (i.e., with the function normalized_mutual_info_score, homogeneity_score, adjusted_rand_score). The purity score was defined as in ref[9].

MATHEMATICAL MODELING

Notation

1. $\chi^{-2}(\nu, \tau^2)$ denotes the scaled inverse Chi-squared distribution.
2. $\otimes$ denotes the Kronecker product.
3. Δ^N is the N-dimensional simplex.

4. I_N is the $N \times N$ identity matrix.
5. $\mathbf{1}_{N \times M}$ is the $N \times M$ matrix of ones.
6. $x_{t_1:t_2}$ denotes the concatenation $\left[x_{t_1} x_{t_1+1}, \dots, x_{t_2}\right]$ where $t_1 < t_2$.

Generative model

Keypoint-MoSeq learns syllables by fitting a switching linear dynamical systems (SLDS) model[13], which decomposes an animal's pose trajectory into a sequence of stereotyped dynamical motifs. In general, SLDS models explain time-series observations $y_1, \dots, y_T$ through a hierarchy of latent states, including continuous states $x_t \in \mathbb{R}^M$ that represent the observations y_t in a low-dimensional space, and discrete states $z_t \in \{1, \dots, N\}$ that govern the dynamics of x_t over time. In keypoint-MoSeq, the discrete states correspond to syllables, the continuous states correspond to pose, and the observations are keypoint coordinates. We further adapted SLDS by (1) including a sticky Hierarchical Dirichlet prior (HDP); (2) excplicitly modeling the animal's location and heading; (3) including a robust (heavy-tailed) observation distribution for keypoints. Below we review SLDS models in general and then describe each of the customizations implemented in keypoint-MoSeq.

Switching linear dynamical systems

The discrete states $z_t \in \{1, \dots, N\}$ are assumed to form a Markov chain, meaning

$$z_{t+1} \mid z_t \sim \text{Cat}\left(\pi_{z_t}\right)$$

where $\pi_i \in \Delta^N$ is the probability of transitioning from discrete state i to each other state. Conditional on the discrete states z_t, the continuous states x_t follow an L-order vector autoregressive process with Gaussian noise. This means that the expected value of each x_t is a linear function of the previous L states $x_{t-L:t-1}$, as shown below,

$$x_t \mid z_t, x_{t-L:t-1} \sim \mathcal{N}\left(A_{z_t} x_{t-L:t-1} + b_{z_t}, Q_{z_t}\right)$$

where $A_i \in \mathbb{R}^{M \times LM}$ is the autoregressive dynamics matrix, $b_i \in \mathbb{R}^M$ is the dynamics bias vector, and $Q_i \in \mathbb{R}^{M \times M}$ is the dynamics noise matrix for each discrete state $i = 1, \dots, N$. The dynamics parameters (A_i, b_i, Q_i) have a matrix normal inverse Wishart (MNIW) prior,

$$[A_i \mid b_i], Q_i \sim \text{MNIW}(\nu_0, S_0, M_0, K_0)$$

where $\nu_0 > M - 1$ is the degrees of freedom, $S_0 \in \mathbb{R}^{M \times M}$ is the prior covariance matrix, $M_0 \in \mathbb{R}^{M \times (LM+1)}$ is the prior mean dynamics matrix, and $K_0 \in \mathbb{R}^{(LM+1) \times (LM+1)}$ is the prior scale matrix. Finally, in the standard formulation of SLDS (which we modify for keypoint data, as described below), each observation $y_t \in \mathbb{R}^D$ is a linear function of x_t plus noise:

$$y_t \mid z_t, x_t \sim \mathcal{N}(Cx_t + d, S)$$

Here we assume that the observation parameters C, d and S do not depend on z_t.

Sticky hierarchical Dirichlet prior

A key feature of depth Moseq[7] is the use of a sticky HDP prior[14] for the transition matrix. In general, HDP priors allow the number of distinct states in a hidden Markov model to be inferred directly from the data. The "sticky" variant of the HDP prior includes an additional hyper-parameter κ that tunes the frequency of self-transitions in the discrete state sequence z_t, and thus the distribution of syllable durations. As in depth MoSeq, we implement a sticky-HDP prior using the weak limit approximation[14], as shown below:

$$\begin{aligned} \beta &\sim \mathrm{Dir}(\gamma/N, \ldots, \gamma/N) \\ \pi_i \mid \beta &\sim \mathrm{Dir}(\alpha\beta_1, \ldots, \alpha\beta_v + \kappa \ldots, \alpha\beta_N) \end{aligned}$$

where κ is being added in the ith position. Here $\beta \in \Delta^N$ is a global vector of augmented syllable transition probabilities, and the hyperparameters γ, α, κ control the sparsity of states, the weight of the sparsity prior, and the bias toward self-transitions respectively.

SLDS for postural dynamics

Keypoint coordinates reflect not only the pose of an animal, but also its location and heading. To disambiguate these factors, we define a canonical, egocentric reference frame in which the postural dynamics are modeled. The canonically aligned poses are then transformed into global coordinates using explicit centroid and heading variables that are learned by the model.

Concretely, let $Y_t \in \mathbb{R}^{K \times D}$ represent the coordinates of K keypoints at time t, where $D \in \{2,3\}$. We define latent variables $v_t \in \mathbb{R}^D$ and $h_t \in [0, 2\pi]$ to represent the animal's centroid and heading angle. We assume that each heading angle h_t has an independent, uniform prior and that the centroid is autocorrelated as follows:

$$\begin{aligned} h_t &\sim \mathrm{Unif}(0, 2\pi) \\ v_t \mid v_{t-1} &\sim \mathcal{N}\left(v_{t-1}, \sigma^2_{\mathrm{loc}}\right) \end{aligned}$$

At each time point t, the pose Y_t is generated via rotation and translation of a centered and oriented pose $\tilde{Y}_t$ that depends on the current continuous latent state x_t:

$$Y_t \quad = \tilde{Y}_t R(h_t) + \mathbf{1}_K v_t^\top \text{ where } \operatorname{vec}(\tilde{Y}_t) \sim \mathcal{N}\big((\Gamma \otimes I_D)(Cx_t + d), S_t\big)$$

where $R(h_t)$ is a matrix that rotates by angle h_t in the xy-plane, and $\Gamma \in R^{K\times(K-1)}$ is defined by the truncated singular value decomposition $\Gamma \Delta \Gamma^\top = I_K - \mathbf{1}_{K\times K}/K$. Note that Γ encodes a linear transformation that isometrically maps $\mathbb{R}^{(K-1)\times D}$ to the set of all centered keypoint arrangements in $\mathbb{R}^{K\times D}$, and thus ensures that $\mathbb{E}(\tilde{Y}_t)$ is always centered[15]. The parameters $C \in \mathbb{R}^{(K-1)D\times M}$, and $d \in \mathbb{R}^{(K-1)D}$ are initialized using principal components analysis (PCA) applied to the transformed keypoint coordinates $\Gamma^T \tilde{Y}_t$. In principle C and d can be adjusted further during model fitting, and we describe the corresponding Gibbs updates in the inference section below. In practice, however, we keep C and d fixed to their initial values when fitting keypoint-MoSeq.

Robust observations

To account for occasional large errors during keypoint tracking, we use the heavy-tailed Student's *t*-distribution, which corresponds to a normal distribution whose variance is itself a random variable. Here, we instantiate the random variances explicitly as a product of two parameters: a baseline variance σ_k for each keypoint and a time-varying scale $s_{t,k}$. We assume:

$$\begin{aligned} \sigma_k^2 &\sim \chi^{-2}(\nu_\sigma, \sigma_0^2) \\ s_{t,k}^2 &\sim \chi^{-2}(\nu_s, s_{0,t,k}) \end{aligned}$$

where $\nu_\sigma > 0$ and $\nu_s > 0$ are degrees of freedom, $\sigma_0^2 > 0$ is a baseline scaling parameter, and $s_{0,t,k} > 0$ is a local scaling parameter, which encodes a prior on the scale of error for each keypoint on each frame. Where possible, we calculated the local scaling parameters as a function of the neural network confidences for each keypoint. The function was calibrated using the empirical relationship between confidence values and error sizes. The overall noise covariance S_t is generated from σ_k and $s_{t,k}$ as follows:

$$S_t = \operatorname{diag}\big(\sigma_1^2 s_{t,1}^2, \dots, \sigma_K^2 s_{t,K}^2\big) \otimes I_D$$

Related work

Keypoint-MoSeq extends the model used in depth MoSeq[7], where a low-dimensional pose trajectory x_t (derived from egocentrically aligned depth videos) is used to fit an autoregressive hidden Markov model with a transition matrix π, autoregressive parameters A_i, b_i, Q_i and discrete states z_t like those described here. Indeed, conditional on x_t, the models for keypoin-MoSeq and depth MoSeq are identical. The main differences are that keypoint-MoSeq treats x_t as a latent variable (i.e. updates it during fitting), includes explicit centroid and heading variables, and uses a robust noise model.

Disambiguating pose from position and heading is a common task in unsupervised behavior algorithms, and researchers have adopted a variety of approaches. VAME[9], for example, isolates pose by centering and aligning data ahead of time, whereas B-SOiD[8] transforms the keypoint data into a vector of relative distances and angles. The statistical pose model GIMBAL[5], on the other hand, introduces latent heading and centroid variables that are inferred simultaneously with the rest of the model. Keypoint-MoSeq adopts this latter approach, which is able to remove spurious correlations between egocentric features that can arise from errors in keypoint localization.

Inference algorithm

Our full model contains latent variables v, h, x, z, s and parameters $A, b, Q, C, d, \sigma, \beta, \pi$. We fit each of these variables – with the exception of C and d – using Gibbs sampling, in which each variable is iteratively resampled from its posterior distribution conditional on the current values of all the other variables. The posterior distributions $P(\pi, \beta \mid z)$ and $P(A, b, Q \mid z, x)$ are unchanged from the original MoSeq paper and will not be be reproduced here (see ref[7], pages 42-44, and note the changes of notation $Q \to \Sigma$, $z \to x$, and $x \to y$). h are described below.

<u>Resampling $P(C, d \mid s, \sigma, x, v, h, Y)$</u>

Let $\tilde{x}_t$ represent x_t with a 1 appended and define

$$\tilde{S}_t \quad = \left(\Gamma^\top \text{diag}\left(\sigma_1^2 s_{t,1}, \dots, \sigma_K^2 s_{t,K}\right)\Gamma\right) \otimes I_D$$

The posterior update is $(C, d) \sim \mathcal{N}(\text{vec}(C, d) \mid \mu_n, \Sigma_n)$ where

$$\Sigma_n \quad = \left(\sigma_C^{-2} I + S_{x,x}\right)^{-1} \quad \text{and} \quad \mu_n = \Sigma_n S_{y,x}$$

with

$$S_{x,x} = \sum_{t=1}^{T} \tilde{x}_t \tilde{x}_t^\top \otimes \Gamma^\top \tilde{S}_t^{-1} \Gamma \otimes I_D \quad \text{and} \quad S_{y,x} = \sum_{t=1}^{T} \left(\tilde{x}_t^\top \otimes \tilde{S}^{-1} \Gamma \otimes I_D\right) \text{vec}\left(\tilde{Y}_t\right)^\top$$

Resampling $P(s \mid C, d, \sigma, x, v, h, Y)$

Each $s_{t,k}$ is conditionally independent with posterior

$$s_{t,k} \mid C, d, \sigma_k, x, Y \sim \chi^{-2}\left(\nu_s + D,\ \left(\nu_s s_0 + \sigma_k^{-2} \parallel (\Gamma(Cx_t + d))_k - \tilde{Y}_{t,k} \parallel^2\right)/(\nu_s + D)\right)$$

Resampling $P(\sigma \mid C, d, s, x, v, h, Y)$

Each σ_k is conditionally independent with posterior

$$\sigma_k^2 \sim \chi^{-2}\left(\nu_\sigma + DT,\ \left(\nu_\sigma \sigma_0^2 + S_y\right)(\nu_\sigma + DT)^{-1}\right)$$

where $S_y = \sum_{t=1}^{N} \parallel \Gamma(Cx_t + d)_k - \tilde{Y}_{t,k} \parallel^2 / s_{t,k}$

Resampling $P(v \mid C, d, \sigma, s, x, h, Y)$

Since the translations $v_1, \ldots, v_T$ form a linear dynamical system, they can be updated by Kalman sampling. The observation potentials have the form $\mathcal{N}(v_t \mid \mu, \gamma^2 I_D)$ where

$$\mu = \sum_k \frac{\gamma_t^2}{\sigma_k^2 s_{t,k}} [Y_{t,k} - R(h_t)^\top \Gamma(Cx_t + d)_k],\quad \frac{1}{\gamma_t^2} = \sum_k \frac{1}{\sigma_k^2 s_{t,k}}$$

Resampling $P(h \mid C, d, \sigma, s, x, v, Y)$

The posterior of h_t is the von-Mises distribution vM(θ, κ) where κ and $\theta \in [0, 2\pi]$ are the unique parameters satisfying $[\kappa\cos(\theta), \kappa\sin(\theta)] = \left[S_{1,1} + S_{2,2}, S_{1,2} - S_{2,1}\right]$ for

$$S = \sum_k \frac{1}{s_{t,k}\sigma_k^2} \Gamma(Cx_t + d)_k \left(Y_{t,k} - v_t\right)^\top$$

Resampling $P(x \mid C, d, \sigma, s, v, h, Y)$

To resample x, we first express its temporal dependencies as a first-order autoregressive process, and then apply Kalman sampling. The change of variables is

$$A' = \begin{bmatrix} & I & & & \\ & & I & & \\ & & & I & \\ A_1 & A_2 & \cdots & A_L & b \end{bmatrix}\ Q' = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & Q \end{bmatrix}\ C' = \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ C & d \end{bmatrix}\ x_t{}' = \begin{bmatrix} x_{t-L+1} \\ \vdots \\ x_t \\ 1 \end{bmatrix}$$

Kalman sampling can then be applied to the sample the conditional distribution,

$$P\left(x'_{1:T} \mid \tilde{Y}_{1:T}\right) \propto \prod_{t=1}^{T} \mathcal{N}\left(x'_t \mid A'^{(z_t)} x'_{t-1}, Q'^{(z_t)}\right) \mathcal{N}\left(\text{vec}\left(\tilde{Y}_t\right) \mid C' x'_t, S_t\right).$$

(Assume x' is left-padded with zeros for negative time indices.)

Hyper-parameters

We used the following hyper-parameter values throughout the paper.

Transition matrix

N	$= 100$
γ	$= 1000$
α	$= 100$
κ	fit to each dataset

Autoregressive process

M	set using PCA explained variance curve
L	$= 3$
ν_0	$= M + 2$
S_0	$= 0.01 I_M$
M_0	$= [0_{M\times(L-1)} \quad I_M \quad 1_{M\times 1}]$
K_0	$= 10 I_{M(L+1)}$

Observation process

σ_0^2	$= 1$
ν_σ	$= 10^5$
ν_s	$= 5$
$s_{0,t,k}$	set based on neural network confidence

Centroid autocorrelation

$\sigma^2_{\text{loc}} = 0.4$

Derivation of Gibbs updates

Derivation of C, d updates

To simply notation, define

$$\tilde{S}_t = \text{diag}\left(\sigma_1^2 s_{t,1}, \ldots, \sigma_K^2 s_{t,K}\right), \quad \tilde{x}_t = (x_t, 1), \quad \tilde{C} = (C, d)$$

The likelihood of the centered and aligned keypoint locations $\tilde{Y}$ can be expanded as follows.

$$P\left(\tilde{Y} \mid \tilde{C}, \tilde{x}, \tilde{S}\right) = \prod_{t=1}^{T} \mathcal{N}\left(\text{vec}\left(\tilde{Y}_t\right) \mid (\Gamma \otimes I_D)\tilde{C}\tilde{x}_t,\ \tilde{S}_t \otimes I_D\right)$$

$$\propto \exp\left[-\frac{1}{2}\sum_{t=1}^{T}\left(\tilde{x}_t^\top \tilde{C}^\top\left(\Gamma^\top \tilde{S}_t^{-1}\Gamma \otimes I_D\right)\tilde{C}\tilde{x}_t - 2\text{vec}\left(\tilde{Y}_t\right)^\top\left(\tilde{S}_t^{-1}\Gamma \otimes I_D\right)\tilde{C}\tilde{x}_t\right)\right]$$

$$\propto \exp\left[-\frac{1}{2}\sum_{t=1}^{T}\left(vec\left(\tilde{C}\right)^\top\left(\tilde{x}_t\tilde{x}_t^\top \otimes \Gamma^\top\tilde{S}_t^{-1}\Gamma \otimes I_D\right)vec\left(\tilde{C}\right)\right.\right.$$
$$\left.\left. -2\text{vec}\left(\tilde{C}\right)^\top\left(\tilde{x}_t^\top \otimes \tilde{S}_t^{-1}\Gamma \otimes I_D\right)\text{vec}\left(\tilde{Y}_t\right)\right)\right]$$

$$\propto \exp\left[-\frac{1}{2}\left(vec\left(\tilde{C}\right)^\top S_{x,x} vec\left(\tilde{C}\right) - 2vec\left(\tilde{C}\right)^\top S_{x,y}\right)\right]$$

where

$$S_{x,x} = \sum_{t=1}^{T} \tilde{x}_t\, \tilde{x}_t^\top \otimes \Gamma^\top \tilde{S}_t^{-1}\Gamma \otimes I_D \quad \text{and} \quad S_{x,y} = \sum_{t=1}^{T}\left(\tilde{x}_t^\top \otimes \tilde{S}^{-1}\Gamma \otimes I_D\right)\text{vec}\left(\tilde{Y}_t\right)$$

Multiplying by the prior $\text{vec}\left(\tilde{C}\right) \sim \mathcal{N}(0, \sigma_C^2 I)$ yields

$$P\left(\tilde{C} \mid \tilde{Y}, \tilde{x}, \tilde{S}\right) \propto \mathcal{N}\left(\text{vec}\left(\tilde{C}\right) \mid \mu_n, \Sigma_n\right)$$

where

$$\Sigma_n = \left(\sigma_C^{-2} I + S_{x,x}\right)^{-1} \quad \text{and} \quad \mu_n = \Sigma_n S_{y,x}$$

Derivation of $\sigma_k, s_{t,k}$ updates

For each time t and keypoint k, let $\bar{Y}_{t,k} = \Gamma(Cx_t + d)$. The likelihood of the centered and aligned keypoint location $\tilde{Y}_{t,k}$ is

$$P\left(\tilde{Y}_{t,k} \mid \bar{Y}_{t,k}, s_{t,k}, \sigma_k\right) = \mathcal{N}\left(\tilde{Y}_{t,k} \mid \bar{Y}_{t,k},\ \sigma_k^2 s_{t,k} I_D\right) \propto \left(\sigma_k^2 s_{t,k}\right)^{-D/2} \exp\left[-\frac{\| \tilde{Y}_{t,k} - \bar{Y}_{t,k} \|^2}{2\sigma_k^2 s_{t,k}}\right]$$

We can then calculate posteriors $P\left(s_{t,k} \mid \sigma_k\right)$ and $P\left(\sigma_k \mid s_{t,k}\right)$ as follows.

$$\begin{aligned} P\left(s_{t,k} \mid \sigma_k, \tilde{Y}_{t,k}, \bar{Y}_{t,k}\right) &\propto \chi^{-1}\left(s_{t,k} \mid \nu_s, s_0\right)\mathcal{N}\left(\tilde{Y}_{t,k} \mid \bar{Y}_{t,k},\ \sigma_k^2 s_{t,k} I_D\right) \\ &\propto s_{t,k}^{-1-(\nu_s+D)/2} \exp\left[\frac{-\nu_s s_0}{2 s_{t,k}} - \frac{\| \tilde{Y}_{t,k} - \bar{Y}_{t,k} \|^2}{2\sigma_k^2 s_{t,k}}\right] \\ &\propto \chi^{-2}\left(s_{t,k} \mid \nu_s + D,\ \left(\nu_s s_0 + \sigma_k^{-2} \| \tilde{Y}_{t,k} - \bar{Y}_{t,k} \|^2\right)(\nu_s + D)^{-1}\right) \end{aligned}$$

$$\begin{aligned} P\left(\sigma_k \mid \{s_{t,k}, \tilde{Y}_{t,k}, \bar{Y}_{t,k}\}_{t=1}^{T}\right) &\propto \chi^{-1}(\sigma_k^2 \mid \nu_\sigma, \sigma_0^2) \prod_{t=1}^{T} \mathcal{N}\left(\tilde{Y}_{t,k} \mid \bar{Y}_{t,k},\ \sigma_k^2 s_{t,k} I_D\right) \\ &\propto \sigma_k^{-2-\nu_\sigma - DT} \exp\left[\frac{-\nu_\sigma \sigma_0^2}{2\sigma_k^2} - \frac{1}{2\sigma_k^2}\sum_{t=1}^{T} \frac{\| \tilde{Y}_{t,k} - \bar{Y}_{t,k} \|^2}{s_{t,k}}\right] \\ &\propto \chi^{-2}\left(\sigma_k^2 \mid \nu_\sigma + DT,\ \left(\nu_\sigma \sigma_0^2 + S_y\right)(\nu_\sigma + DT)^{-1}\right) \end{aligned}$$

where $S_y = \sum_t \| \tilde{Y}_{t,k} - \bar{Y}_{t,k} \|^2 / s_{t,k}$

Derivation of v_t update

We assume an improper uniform prior on v_t, hence

$$\begin{aligned} P(v_t \mid Y_t) &\propto P(Y_t \mid v_t) P(v_t) \propto P(Y_t \mid v_t) \\ &\propto \mathcal{N}\left(vec\left((Y_t - \mathbf{1}_K v_t^\top) R(h_t)^\top\right) \mid \Gamma(Cx_t + d),\ S_t\right) \\ &= \prod_k \mathcal{N}\left(R(h_t)\left(Y_{t,k} - v_t\right) \mid \Gamma(Cx_t + d)_k,\ s_{t,k}\sigma_k^2 I_D\right) \\ &= \prod_k \mathcal{N}\left(v_t \mid Y_{t,k} - R(h_t)^\top \Gamma(Cx_t + d)_k,\ s_{t,k}\sigma_k^2 I_D\right) \\ &= \mathcal{N}(v_t \mid \mu_t, \gamma_t^2 I_D) \end{aligned}$$

where

$$\mu = \sum_k \frac{\gamma_t^2}{\sigma_k^2 s_{t,k}}\left(Y_{t,k} - R(h_t)^\top \Gamma(Cx_t + d)_k\right), \quad \frac{1}{\gamma_t^2} = \sum_k \frac{1}{\sigma_k^2 s_{t,k}}$$

Derivation of h_t update

We assume a proper uniform prior on h_t, hence

$$\begin{aligned}
P(h_t \mid Y_t) &\propto P(Y_t \mid h_t)P(h_t) \propto P(Y_t \mid h_t) \\
&\propto \exp\left[\sum_k \frac{\left(Y_{t,k} - v_t\right)^{\top} R(h_t)\Gamma(Cx_t + d)_k}{s_{t,k}\sigma_k^2}\right] \\
&= \exp\left[\frac{\mathrm{tr}\left[R(h_t)\Gamma(Cx_t + d)_k\left(Y_{t,k} - v_t\right)^{\top}\right]}{s_{t,k}\sigma_k^2}\right] \\
&\propto \exp\mathrm{tr}[R(h_t)S] \quad \text{where} \quad S = \sum_k \Gamma\,(Cx_t + d)_k\left(Y_{t,k} - v_t\right)^{\top}/\left(s_{t,k}\sigma_k^2\right) \\
&\propto \exp\left[\cos(h_t)\left(S_{1,1} + S_{2,2}\right) + \sin(h_t)\left(S_{1,2} - S_{2,1}\right)\right]
\end{aligned}$$

Let $[\kappa\cos(\theta), \kappa\sin(\theta)]$ represent $\left[S_{1,1} + S_{2,2}, S_{1,2} - S_{2,1}\right]$ in polar coordinates. Then

$$\begin{aligned}
P(Y_t \mid h_t) &\propto \exp[\kappa\cos(h_t)\cos(\theta) + \sin(h_t)\sin(\theta)] \\
&= \exp[\kappa\cos(h_t - \theta)] \propto \mathrm{vM}(h_t \mid \theta, \kappa)
\end{aligned}$$

Supplemental References

1 Markowitz, J. E. *et al.* Spontaneous behaviour is structured by reinforcement without explicit reward. *Nature* **614**, 108-117 (2023). https://doi.org:10.1038/s41586-022-05611-2

2 Lin, S. *et al.* Characterizing the structure of mouse behavior using Motion Sequencing. (2022). https://doi.org:10.48550/ARXIV.2211.08497

3 Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N. & Liang, J. in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support.* (eds Danail Stoyanov *et al.*) 3-11 (Springer International Publishing).

4 Sun, K., Xiao, B., Liu, D. & Wang, J. in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 5686-5696.

5 Zhang, L., Dunn, T., Marshall, J., Olveczky, B. & Linderman, S. in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics* Vol. 130 (eds Banerjee Arindam & Fukumizu Kenji) 2800--2808 (PMLR, Proceedings of Machine Learning Research, 2021).

6 Nath, T. *et al.* Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature Protocols* **14**, 2152-2176 (2019). https://doi.org:10.1038/s41596-019-0176-0

7 Wiltschko, A. B. *et al.* Mapping Sub-Second Structure in Mouse Behavior. *Neuron* **88**, 1121-1135 (2015). https://doi.org:10.1016/j.neuron.2015.11.031

8 Hsu, A. I. & Yttri, E. A. B-SOiD, an open-source unsupervised algorithm for identification and fast prediction of behaviors. *Nature Communications* **12**, 5188 (2021). https://doi.org:10.1038/s41467-021-25420-x PMID - 34465784

9 Luxem, K. *et al.* Identifying behavioral structure from deep variational embeddings of animal motion. *Commun Biol* **5**, 1267 (2022). https://doi.org:10.1038/s42003-022-04080-7

10 Berman, G. J., Choi, D. M., Bialek, W. & Shaevitz, J. W. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of the Royal Society, Interface / the Royal Society* **11** (2014). https://doi.org:papers3://publication/doi/10.1098/rsif.2014.0672

11 Bohnslav, J. P. *et al.* DeepEthogram, a machine learning pipeline for supervised behavior classification from raw pixels. *eLife* **10**, e63377 (2021). https://doi.org:10.7554/eLife.63377

12 Sun, J. J. *et al.* Caltech Mouse Social Interactions (CalMS21) Dataset. (2021). https://doi.org:10.22002/D1.1991

13 Ackerson, G. & Fu, K. On state estimation in switching environments. *IEEE Transactions on Automatic Control* **15**, 10-17 (1970). https://doi.org:10.1109/TAC.1970.1099359

14 Fox, E. B., Sudderth, E. B., Jordan, M. I. & Willsky, A. S. in *Proceedings of the 25th International Conference on Machine Learning* 312–319 (Association for Computing Machinery, 2008).

15 Andreella, A. & Finos, L. Procrustes Analysis for High-Dimensional Data. *Psychometrika* **87**, 1422-1438 (2022). https://doi.org:10.1007/s11336-022-09859-5