Supplemental Data

Figure S1. Image extraction workflow. Related to Figure 1. Workflow (shown here with text descriptions of each step **above** and graphics corresponding to each step **below**) used to convert depth images of freely-behaving mice into mouse pose dynamics data in which each mouse is aligned along the axis of its spine (see Movie S1 for both the raw input data and the product of this process). This process is described in detail in the Supplemental Experimental Procedure. In brief, after recording depth data within a behavioral apparatus (**A**), background values are computed (**B**) and missing pixel values are imputed (**C**). The mouse is then resampled into real-world coordinates to correct for parallax artifacts (**D**-**E**), and the background contributed by the apparatus is subtracted away (**F**). These images are then filtered, (**G**) and the image of the mouse is identified and extracted as a region of interest (**H**-**I**). The unrotated image is then aligned longitudinally based on fitting an ellipse to the boundary of the image (the major axis of which we refer to as "the axis of its spine) (**J**), enabling quantitative analysis of egocentric pose dynamics while the animal freely translates in space. This pre-processed imaging data is then subject to wavelet decomposition and PCA for dimensionality reduction before submission to the modeling algorithms.

Figure S2. Block structure, autocorrelation and compressibility in mouse depth imaging data. Related to Figure 1. A. Block structure is present in random projections data, spine data and raw pixel data derived from aligned mouse pose dynamics data (random projections and spines are depicted as in Fig. 1A, raw data in bottom panels are heatmapped based upon imaging pixel value; values are sorted by mean height, and rarely-used pixels are omitted). Each dimension of random projections data uses all pixels from the original mouse image. Live mice

exhibit significant block structure in imaging data (left panels), while dead mice do not (right panels). Smoothing refers to the modest median filtration applied during pre-processing, which does not visibly affect the structure of the data (Supplemental Experimental Methods). **B.** Dimensionality reduction does not significantly affect autocorrelation structure or spectral content of mouse pose dynamics data, nor does the particular experimental setup. The result of the Wiener filtering analysis (**left**) is similar for both a 10-dimensional PCA and a random projection representation of depth data, as well as across experimental conditions (OFA = open field assay, TMT Odor = square box with the fox odor TMT in one quadrant, Blank Odor = square box without introduced odorants, RorB = OFA data with mice mutant, heterozygous or wild-type at the Ror1β locus, Optogenetics = optogenetic experiment in the OFA with unilateral motor cortex implantation of an optical fiber). The autocorrelation of random projections and PCA data (**right**), both of which represent the same depth data, exhibit approximately the same falloff, demonstrating that data compression does not influence fine-timescale autocorrelation structure in the imaging data. The particular arena in which the mouse was recorded, or the experimental manipulation to which the mouse was subjected also does not affect the autocorrelation structure, indicating it is a core feature of our data. Note that the slightly different autocorrelation structure observed in the optogenetics data can be explained by the presence of an optical fiber implanted in the head of the animal, which alters the imaged pose dynamics of the animal. **C.** This correlation structure is not observed if mouse poses evolve as if taking a Levy flight (**left**) or random walk (**right**), suggesting that live mice express a specific sub-second autocorrelation structure. The alpha parameter and sigma parameter control the rate of dispersion for data generated from the Levy flight and Brownian motion processes, respectively. **D.** The

first 10 principal components capture 88 percent of the variance in raw mouse imaging data. This number of dimensions was used for data analysis in the AR-HMM.

Figure S3. Additional template-matched trajectories in pose space. Related to Figure 2. Scanning the behavioral data using a template matching method identified additional instances in which given stereotyped motifs were reused (with time proceeding from blue to red). Six additional templates and their nine closest matched trajectories are plotted in the first two dimensions of principal components. The complete pose data in principal component space is represented as a background density plot. Note that 10 PCs were used to identify examples matching the seed templates, although only 2 PCs are depicted here for clarity.

Figure S4. **Modeling mouse behavior**. **Related to Figure 2 and Supplemental Experimental Procedures**. **A**. Sketch of the part of the data processing pipeline that includes model fitting. The mouse is first identified, extracted and aligned from depth camera video frames along the major axis of the mouse, which approximates its spine. Parallax artifacts are removed. The tracked mouse images are then subjected to a wavelet decomposition, and then reduced to a principal component representation, yielding a multidimensional time series. This principal component time series is then modeled with one of a family of models, which is then fit using Gibbs sampling iterations; this procedure segments the principal component time series into reused behavioral modules. **B.** The workflow for fitting behavioral data with the AR-HMM, the best-performing of the family of models composed to represent different hypotheses about the underlying structure of behavior. The input to the workflow extracted and aligned mouse images, and the output is a label sequence equal in length to the original data timeseries that assigns a

behavioral module identity for every recorded frame of behavioral data. This process is described in detail in the Supplemental Experimental Procedures. (i) First, after pre-processing image data are compressed from 3600 to 10 dimensions via wavelet decomposition and PCA. (ii) Data are randomly split into "train" and "test" segments (in a 70:30 ratio), in order to evaluate the performance of the model on unseen data, and to guard against overfitting. (iii) The parameters of the model are randomly initialized (parameters describing each module's characteristic trajectory through pose space, duration distribution, and transition statistics relative to other modules), and fit iteratively with Gibbs sampling. Gibbs sampling alternates between several updates: the algorithm first attempts to segment the imaging data into modules given a fixed set of transition statistics and a fixed description of the AR parameters that describe any given module, and then the algorithm switches to fixing the segmentation and updating the transition matrix and the AR parameters. Filled circles represent fixed parameters, while open circles represent parameters that are varied in a particular iteration. (iv) The product of this process is the parameters described above: "AR Matrices," the AR parameters that describe the specific trajectory through pose space that defines any given behavioral module, "Duration Distributions," the parameters describing the duration distribution for each behavioral module (in model classes in which this distribution is explicitly modeled), "Transition Matrix," the probability that any given module will precede or follow any other module, and "State Labels," the behavioral module inferred for each frame of 3D video data. (v) The quality of the fit in terms of predicting behavior is evaluated using the likelihood of data that was held out from training. **C**. A schematic representation of the aspects of behavior captured by the AR-HMM. The AR-HMM takes in a sequence of mouse behavioral data (fed into the model as 10 PCs, which are derived from pixel data), and generates a sequence of re-used labels over time

("Labels"), such that every moment of the mouse's behavior is associated with a single behavioral module. Each behavioral module is internally modeled as an autoregressive process that can be visualized as a trajectory through pose space ("AR Process 1," AR Process 2," etc.). The transition probabilities between modules are represented in a transition matrix, which may be represented as a state map, where each behavioral module is a node, and the probability of transition between two nodes is represented by the thickness of a directed arrow between those nodes ("Transition Probabilities"). **D**. The AR-HMM outperforms alternative models as measured by next-frame average held-out log likelihood. **E.** Data generated from a fit AR-HMM model are qualitatively similar to PCs derived directly from imaging data.

Figure S5. Characterizing behavioral modules identified by the AR-HMM. Related to Figure 2 and Supplemental Experimental Procedures. A. Duration distributions for changepoint-identified blocks and AR-HMM-identified modules are similar. **B.** Shuffling behavioral data at fast timescales lowers AR-HMM performance, as measured by the ratio of held-out likelihood before and after shuffling; destroying temporal information at a timescale longer than 500 ms has essentially no effect on model performance, while shuffling at faster timescales catastrophically prevents effective behavioral prediction. **C.** Data instances drawn from different behavioral modules trace distinct paths through PCA space. Six separate representative behavioral modules are plotted in the first two principal components, overlaid on a density indicating the distribution of all recorded poses. Note that the data represented here are plotted in 2D; pose dynamics are significantly more separated in 10D (see Fig 2D). **D.** 95 percent of frames were explained by only 51 behavioral modules; 99 percent of frames were explained by 65 behavioral modules in the open field dataset (error bars indicate ± S.E. calculated with

bootstrap samples). **E.** Modules (X axis) sorted by usage (Y axis) with Bayesian credible intervals indicated as error bars, which are smaller than SEs computed with the nonparametric bootstrap (see Fig. 2). **F.** Module interconnectivity is sparse. With no thresholding, each module is interconnected with on average 16.85±0.95 other modules, with the rate of interconnectivity falling sharply with even modest thresholding (Y axis indicates interconnectivity, ± SE calculated via the nonparametric bootstrap, X axis indicates thresholding applied to transition probabilities). This is consistent with sparse temporal interconnectivity between behavioral modules. These transition probabilities are distinct from bigram probabilities (e.g., Fig. 3B & 4C); bigram probabilities represent the frequency with which one module occurs after another as a proportion of all pairs of temporally adjacent modules. Transition probabilities, in contrast, represent the frequency with which one module occurs after the other as a proportion of all instances of only the preceding module.

Figure S6. Subjective and objective analysis of AR-HMM identified behavioral modules. Related to Fig. 4. A. Histogram depicting the number of modules that fall into one of five coarse labels of behavior, as scored by a human observer; this coarse subjective description is intended to offer a sense of the distribution of behaviors parsed by the AR-HMM. Note that these module descriptors are completely subjective, and applied post-hoc after modeling. Furthermore, inspection revealed that the modules within each broad category could be further distinguished from the others within that category; the "rear" category, for example, included a wide variety of behaviors in which the mouse's head was transiently or perdurantly elevated, including true rears, elevated pointing behaviors, head-bobs, and sniffs. **B**. Histogram depicting the average

velocity of the modules that were interconnected after TMT exposure (red), which individually and collectively encode "freezing," compared to all other modules in the dataset.

Figure S7. Sensitivity analysis of model-free parameters. Related to Supplemental Experimental Procedures. A. To denoise data from the Kinect a filtering approach was used in which a median filter was applied iteratively both in space and in time; this approach has been shown to effectively maintain data structure while smoothing away noise (see Supplemental Experimental Procedures). To determine filter settings, we imaged dead mice that were differentially posed in rigor mortis; ideal filter settings would distinguish mice that were posed differently, but be unable to distinguish data from the same mouse. Filter settings are indicated as ((pixels), (frames)) with the numbers within each parenthesis referring to the iterative settings for each round of filtering. To assess filter performance, a within/between pose correlation ratio (Y axis) was computed, in which the mean spatial correlation for all frames of the same pose was divided by the mean spatial correlation for all frames of different poses. This revealed that light filtering (with settings ((3), (3,5))) optimized discriminability in the data. **B.** Optimal parameters for changepoints analysis were identified by grid scanning. By maximizing the number of changepoints identified in a live mouse, while fixing the number in a dead mouse to be zero, clear optimal values were identified via grid scanning for sigma and H (left two panels). This changepoint ratio was not highly sensitive to K; a setting of 48 (at the observed maximum) was therefore chosen.

Movie S1. Related to Fig. 1A. Imaging the three-dimensional pose dynamics of freely-behaving mice in the open field with depth cameras. A depth camera placed above the arena

captures the three-dimensional shape of the mouse at 30 frames-per-second as it freely behaves (**right,** height is colormapped as indicated). These raw imaging data are pre-processed (filtered, background subtracted and parallax corrected) and submitted to a machine learning algorithm that aligns the animal along the axis of its spine, thereby enabling quantitative measurements of how its pose dynamics evolve over time (**left**, see Experimental Procedures for details). Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S2. Related to Fig. 1D. Changepoints analysis identifies approximate boundaries between behavioral motifs. Three-dimensional movie of a freely-behaving mouse, with changepoints indicated on upper left. Periods of behavior between the changepoints, which identifies approximate boundaries between blocks, appear to capture behavioral motifs. Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S3. Related to Fig. 2B, 2C. The AR-HMM identifies a stereotyped and reused behavioral module that encodes low rearing. Three separate 3D behavior movies (**top left, top right, bottom left**) taken from a single mouse in a single experiment, each depicting a separate instance in which the AR-HMM identified a particular sequence of frames as encoding the same behavioral module. In these movies, the ongoing expression of the module is indicated by a dot being placed over the mouse; imaging frames before the dot onset and after the dot offset are associated with different behavioral modules that are distinct in all three movies, as is the allocentric position of the mouse within the arena. Nevertheless, during expression of the module (i.e., when the dot is over the animal), the mouse in all three examples emits a stereotyped motif of three-dimensional motion that can be identified in these movies as a low rear. By collating and

overlaying 20 of these individual movies (high-likelihood examples taken from 20 different experiments using 20 individual mice) into a single movie (**lower right**), the stereotyped nature of this behavioral module is clear. Note that this "crowd" movie is timelocked such that all the modules are expressed at approximately the same time during the movie; this timelocking was achieved by aligning the middle frame of the module across all examples. Despite this timelocking, the duration of each instance of the behavioral module is slightly different; this is because the AR-HMM defines a unique duration distribution for every behavioral module, rather than a single duration. Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S4. Related to Fig. 2C. Behavioral modules are distinct, stereotyped, and reused motifs of behavior. "Crowd" movies (see legend for Movie S3) depicting 10 distinct behavioral modules automatically identified by the AR-HMM, each labeled with a descriptor; three of these are from the trigram depicted in Fig. 2. Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S5. Related to Fig. S4E, 2D. The synthetic output of the AR-HMM appears similar to the pose dynamics data exhibited by real mice. A "dream" (left) and real (right) mouse in aligned pose space during normal exploration. The images on the left were generated by training an AR-HMM on open field behavioral data and then, after training, having the model emit synthetic data representing its best estimate of mouse behavior (Supplemental Experimental Procedures). Note that there is some sensor noise apparent in the real mouse (right) that is not modeled by the AR-HMM, but that otherwise these movies are visually very similar.

Movie S6. Related to Fig. 2B. Module sequences encode exploratory behaviors. Three-dimensional imaging of three separate examples of the sequence of three modules ("walk," "pause" and "low rear") from Fig. 2 being expressed by mice during free behavior. Note that, in this case, when the dot comes over the animal (see legend for Movie S3 for details) the sequence is being expressed, rather than a single module. Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S7. Related to Fig. 3D. Modules encoding wall-hugging behaviors. "Crowd" movie (see legend to Movie S3 for details) of an open-field specific behavioral module (**left**) encoding a behavior in which mice locomote while hugging the walls of the apparatus, referred to as thigmotaxis. In this case, the module can be seen to encode locomotion with a body habitus that is slightly curved to match the curve in the circular walls of the arena. A similar thigmotaxis module can be seen in the square (**right**); in this case, however, the syllable is also used during straight patterns of locomotion in both the square box and the circular open field arena. Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S8. Related to Fig. 4E. TMT-regulated behavioral sequences encode freezing behavior. "Crowd" movies of three behavioral modules that are placed into series after exposure to the aversive fox odor TMT (see legend to Movie S3 for details); these trigrams reveal that this sequence of behavioral modules encodes freezing behavior. Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S9. Related to Fig. 4D,4E. TMT regulates where behavioral modules are expressed within an experimental apparatus. "Crowd" movie (see legend to Movie S3 for details) of two separate behavioral modules whose usage is not altered by the aversive fox odor TMT, but whose pattern of allocentric deployment is changed. An investigatory sniffing module is expressed near the TMT quadrant (**left**), and a pausing module is expressed away from the TMT quadrant (**right**). Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S10. Related to Fig. 5A, 5C. ***Ror1β*** **mice express a waddling module.** "Crowd" movie (see legend to Movie S3 for details) of a *Ror1β*−specific behavioral module that encodes a waddling gait; note that upon the expression of the module the animal's hindquarters waggle during locomotion. Color scale bar indicates the height of any given pixel above the floor of the arena (mm).

Movie S11. Related to Fig. 6B. Optogenetically-induced spinning behavior. "Crowd" movie (see legend to Movie S3 for details) of the most upregulated behavioral module in response to optogenetic stimulation of the left motor cortex. This module is not expressed during normal behavior, but encodes pathological spinning behaviors that are associated with strong unilateral stimulation of corticostriatal neurons.

Supplemental Experimental Procedures

Methods Summary

Experiments, unless otherwise noted, were performed on C57/BL6 male mice, aged 6-8 weeks, on the dark cycle of a reverse 12 hour light / 12 hour dark cycle. All mice were videotaped with a Microsoft Kinect depth camera and custom recording software. Recorded depth video was subjected to custom algorithms to extract, parallax correct and align the 3D image of the mouse, and then the time-series data were dimensionally compressed using a wavelet transformation and PCA before being subjected to both model-free and model-based analysis. Autocorrelation analysis was performed using correlation distance between time-lagged data points to correct for sensor-specific noise. Spectral analysis was carried out using the Welch periodogram method, and approximate changepoints were identified using a filtered derivative algorithm. Template matching was achieved using a minimal Euclidean distance criterion. Established Bayesian time-series modeling methods were modified to identify behavioral modules. The core AR-HMM was built from three components: 1) an observation distribution, describing the fast-timescale pose dynamics of each behavioral module; 2) a duration distribution, to allow variability in the length of emission of each behavioral module; and 3) a transition distribution (when relevant), which summarizes the interconnectivity over time of identified behavioral modules. Each behavioral module has its own distinct pose dynamics, its own characteristic duration distribution, and its own transition structure in relation to all other behavioral modules. Models were fit using Markov Chain Monte Carlo methods such as Gibbs sampling. The quality of fit was evaluated by testing the model on previously held-out data, which allows direct comparison between alternative model constructions.

Statistical significance of differences in behavioral module usage and transition usage were evaluated by computing both bootstrap samples over the data (on a per-mouse basis) and Bayesian credible intervals. Group comparisons were performed using the Hotelling t-squared statistic, single hypothesis tests using the Wald test. Both tests were adjusted with the Holm-Bonferroni step-down procedure to correct for multiple comparisons.

Mouse Strains, Housing and Habituation

Unless otherwise noted, all experiments were performed on 6-8 week old C57/BL6 males (Jackson Laboratories). Mice from the *Ror1β* and *Rbp4-Cre* strains (Jackson Laboratories) were habituated and tested identically to the reference C57/BL6 mice. Mice were brought into our colony at 5 weeks of age, where they were group-housed for one week in a reverse 12 hours light/12 hours dark cycle. On the day of testing, mice were brought into the laboratory in a light-tight container, where they were habituated in darkness for 20 minutes before testing.

Behavioral Assays: Innate Exploration

For the open field assay (OFA), mice were habituated as noted above and then placed in the middle of a circular 17" diameter enclosure with 15"-high walls (US Plastics), immediately after which 3D video recording was begun. Mice were allowed to freely explore the enclosure for the 20 minute experimental period. Mice whose behavior was assessed in a square box were handled identically to the OFA.

Behavioral Assays: Stimulus-Driven Innate Behaviors

To assess innate behavioral responses to volatile odorants, an odor delivery system was developed that spatially isolates odors in specific quadrants of a square box. Each 12” x 12” box was constructed of ¼” black matte acrylic (Altech Plastics), with 3/4” holes patterning the bottom of the box in a cross formation, and a 1/16" thick glass cover (Tru Vue). These holes were tapped and connected via PTFE tubing to a vacuum manifold (Sigma Aldrich) that provides negative pressure to isolate odors within quadrants. Odor was injected into the box through ½” NPT-⅜” pipe-fittings (Cole-Parmer). Filtered air (1.0 L/min) was blown over odorant-soaked blotting paper (VWR) placed at the bottom of Vacutainer syringe vials (Covidien). The odorized airstream was then passed through corrugated PTFE tubing (Zeus) into one of the four pipe-fittings in a corner of the odor box. The ability of the odor box to isolate odors within specified quadrants was verified by visualizing vaporized odor or smoke through sheet illumination of the box with a low-power handheld HeNe laser. This approach allowed us to tune the vacuum flow and odor flow rates to achieve odor isolation, which was verified using a photoionization device (Aurora Scientific). To eliminate the possibility of cross contamination between experiments, the odor boxes were soaked in a 1% Alconox solution overnight, then thoroughly cleaned with a 70% ethanol solution. Mice were habituated to the experimental room for 30 minutes before the initiation of the experiment. Under control conditions, dipropylene glycol with air (1.0 L/min) was delivered to each of the four corners of the apparatus before a single mouse was placed in the center of the box and allowed to freely explore while 3D video records were acquired for 20 minutes. The same cohort of animals was tested for odor responses by subsequently repeating the experiment with odorized air delivered to one of the four quadrants. All 3D video recordings were performed in total darkness. TMT (Pherotech) was used at 5% concentration.

Behavioral Assays: Optogenetics

Four adult male *Rbp4*-Cre (The Jackson Laboratory) mice were anesthetized with 1.5% isoflurane and placed in a stereotaxic frame (Leica). Microinjection pipettes (O.D. 10-15 μm) were inserted into the left motor cortex (coordinates from Bregma: 0.5 AP, -1 ML, 0.60 DV). 0.5 μl of AAV5.EF1a.DIO.hChR2(H134R)-eYFP.WPRE.hGH (~10^{12} infectious units/mL, Penn Vector Core) was injected in each mouse over 10 minutes followed by an additional 10 minutes to allow diffusion of viral particles away from the injection site. After the injection, a bare optical fiber with a zirconia ferrule (O.D. 200 μm, 0.37 numerical aperture) was inserted 100 μm above the injection site and secured to the skull with acrylic cement (Lang). Twenty-eight days following the viral injection, mice were placed in a circular arena and the optical implant was coupled to a laser pump (488 nm, CrystaLaser) via a patch-chord and a rotary joint (Doric Lenses). The laser was directly controlled from a PC. After allowing the mouse 20 minutes of familiarization to the arena, optogenetic stimuli were delivered. The laser power, the pulse width, the inter-pulse interval and the inter-train interval were controlled by custom-made software (NI Labview). Each train of laser pulses consisted of 30 pulses (pulse width: 50 ms) at 15 Hz. The interval between successive trains was set to 18 seconds. 50 trains were delivered for each laser intensity. The animal was progressively exposed to higher laser intensities over the course of the experiment.

Camera Setup and Initialization

Mice were tracked in 3D using a Kinect for Windows (Microsoft). A boom tripod (Manfrotto) was used to suspend the camera above the recording arena, affording a stable top-down view of the mouse. The Kinect has a minimum working distance (in Near Mode) of 0.5

meters; by quantitating the number of missing depth pixels within an imaged field, the optimal sensor position was found to be between 0.6 and 0.75 meters depending on ambient light conditions and assay material. Thermal effects can cause degradation in the signal coming off the sensor; after each recording session, the camera was therefore unplugged from its power source to allow the sensor to deactivate and cool. The Kinect sensor has a small indicator LED that was masked using electrical tape so as to not emit spurious illumination.

Data Acquisition

Data from the Kinect was sent to an acquisition computer (hand-assembled, 16GB RAM, Intel i7, 512GB SSD) via USB. Custom Matlab software was used to interface the Kinect via the official Microsoft .NET API that retrieves depth frames at a rate of 30 frames per second and saves them in raw binary format (16-bit signed integers), along with the timestamp of each frame, in milliseconds, to an external hard-drive. After acquisition, we indicate a region-of-interest (ROI) delimiting the boundaries of the experimental arena using Matlab GUI tools, and save the polygon specifying the ROI containing the arena.

Data Preprocessing

All data analysis performed with Python scientific computing tools and OpenCV for image analysis (Pedregosa et al., 2011; Perez and Granger, 2007; van der Walt et al., 2011) on r3.8xlarge EC2 virtual machines running an AMI customized from a base Ubuntu 14.04 Linux AMI. Tracking the evolution of an imaged mouse's pose over time requires identifying the mouse within a given video sequence, segmenting the mouse from the background (in this case the apparatus the mouse is exploring), orienting the isolated image of the mouse along the axis of

its spine, correcting the image for perspective distortions, and then compressing the image for processing by the model.

To isolate our analysis to the experimental arena in which the mouse is behaving, a region-of-interest (ROI) was identified for further analysis that was manually traced along the outside edge of any imaged arena; pixels outside this ROI were by default set to zero to prevent spurious object detection. The raw imaging data were filtered with an iterative median filter (Arias-Castro and Donoho, 2007), which is well suited to removing correlated noise from the sensor within the Kinect. Filter settings were identified that maximized between-pose discriminability while minimizing within-pose differences by using dead mice in rigor mortis that were placed into distinct poses (Fig. S7A); based upon this analysis, a three pixel median filter over space, and then a three-then-five frame iterative median filter over time was applied to the imaging data.

To subtract the background image of the arena, the median value of the first 30 seconds of data from any imaging stream were taken, and subtracted from all video frames; any spurious values less than zero were by default reset to zero. The image was then binarized and any objects that did not survive three iterations of morphological opening were eliminated. The mouse was defined as the largest object within the arena that survived the subtraction and masking procedures. The centroid of the mouse was identified as the center-of-mass of the preprocessed image; an ellipse was fit to its contour to detect its overall orientation. In order to orient the mouse along the axis of the spine, a random forest classifier was trained on a set of manually oriented mouse images to facilitate automated identification of the head and tail. Although this algorithm was nearly always correct, its output was manually supervised to ensure accuracy. Note that this procedure does not directly identify the spine, but that the ellipse-based alignment

procedure has the effect of aligning the animal along this anatomic axis. During episodes when the mouse was reared directly upwards (and therefore appears as a circle against the sensor, which is orthogonal to the axis of the mouse's spine under these conditions) neither a human nor our algorithm can identify the spinal axis; because we model pose dynamics based upon dimensionally-reduced representations of the pixels (rather than extracted scalar features, like orientation or length), the presence of this symmetry does not adversely affect model performance, and synthetic data generated from the model appropriately captures these rearing episodes without "flipping" the mouse along its spinal axis.

The position of the mouse within the arena enabled extraction of allocentric data about the mouse, including the centroid, head and tail positions of the mouse, orientation, length, width, height, and each of their first derivatives with respect to time. Critically, none of these scalar parameters are fed to the model; we obtain these data for the sake of convenience and to understand how the egocentric behaviors captured by the model might relate to allocentric parameters like position or velocity.

To correct perspective distortion in the X and Y axes, a tuple of (x,y,z) coordinates was generated for each pixel in real-world coordinates, and then those coordinates were resampled to fall on an even grid in the (x,y) plane using Delaunay triangulation. This procedure corrected for the slight differences in the appearance of the mouse when at the left or right edges of the arena; these differences derive from parallax effects caused by imaging the animal from a single fixed vantage point from above, but are normalized by the interpolation procedure. All extracted images and scalars were saved in HDF5 format. The extraction pre-processing which transforms raw data into the extracted HDF5 format runs at approximately real-time (e.g. 20 minutes of

preprocessing is required for a 20 minute recording session) on a 32-core r3.8xlarge EC2 instance.

The full extracted image size of a mouse covered an area of 120 mm^2, where each pixel is 2 mm on a side; each frame to be submitted for analysis contained 3,600 16-bit integers (which were cast to floating point before any image analysis) in a 60x60 array. The information captured in this image was often either highly correlated (neighboring pixels) or uninformative (pixels on the border of the image that never contain the mouse's body). To both reduce redundant information and make modeling computationally tractable, each image was dimensionally reduced. A five-level wavelet decomposition was first applied, thereby transforming the image into a representation in which each dimension captured and pooled information at a single spatial scale; in this transformation, some dimensions coded explicitly for fine edges on the scale of a few millimeters, while others encoded broad changes over spatial scales of centimeters (Mallat, 1989). This wavelet decomposition expanded the dimensionality of the image; to reduce this dimensionality principal components analysis was applied to these vectors, and projected the wavelet coefficients into ten dimensions, which captures 88% of total variance (Fig. S2D) and which were submitted to our modeling algorithms. Principal components were built using a canonical dataset of 25 C57 BL/6 mice, aged 6 weeks, recorded for 20 minutes each in the OFA, and all datasets were projected into this common pose space.

For some model-free depictions of behavioral data, the random projections technique was used to reduce the data dimensionality. This approach also has the benefit of representationally normalizing the data to changes in the size of the mouse on the sensor. For example, when a mouse rears up the number of pixels that describe the mouse changes drastically, leading to obvious striations in the raw pixel data over time that may visually overstate the overall change

in the behavior of the mouse. Because, as described below, each random projection dimension takes into account all of the pixels within the 60x60 array, representations of the random projections data are not dominated by these effects, enabling more holistic inspection of changes in the data over time. Random projections is an approach that produces new dimensions derived from an original signal, with dimensionality D_orig, by randomly weighting each original dimension, and then summing each dimension according to that weighting, producing a single number per data point. This procedure is repeated several times, with new random weightings, to produce a set of "randomly projected" dimensions. The Johnson–Lindenstrauss lemma shows that distances between points in the original dataset with dimensionality D_orig are preserved in the randomly projected dimensions, D_proj, where D_proj < D_orig. As random projections is primarily employed as a visualization aid, we set D_proj << D_orig, where D_proj=300, D_orig=3600. For visualization purposes we do not sort the random projections dimensions in any way.

Model-Free Data Analysis Methods

Standard methods for computing autocorrelation were modified, as sensor-specific noise results in a monotonically decreasing autocorrelogram, even for a mouse that is posed in rigor mortis. Average correlation of all 10 dimensions of the mouse's pose data was therefore used as the comparator between time-lagged versions of the time-series signal in question, resulting in a nearly flat autocorrelation function of value ~1.0 for a dead mouse, and a declining autocorrelation function for a behaving mouse. The recording of the dead mouse was performed after the animal had entered rigor mortis; the posed dead mouse was placed in the open field arena and recorded for 5 minutes. The rate at which this autocorrelogram declines in a behaving

mouse may be characterized as a time-constant, tau, of an exponentially decaying curve. Tau was fit using the Levenberg-Marquardt algorithm (non-linear least squares) using the SciPy optimization package.

Power-spectral density (PSD) analysis was performed on behavioral data to further analyze its time domain structure (Welch, 1967) using a Wiener filter (Wiener, 2013), and implemented by taking the ratio of the PSD of a behaving mouse over the PSD of a dead mouse. The PSD was computed using the Welch periodogram method, which takes the average PSD over a sliding window across the entire signal; a 33-second window was used, sliding in 0.8 second increments.

Plotting the mouse depth data over time (as either raw data or as random projections) yields obvious striations, each a potential boundary between blocks or modules. To automate the identification of these approximate boundaries between blocks, a simple changepoint identification technique was used, called the filtered derivative algorithm (Basseville and Nikiforov, 1993). Briefly, the algorithm calculated the derivative of the per-frame unit-normalized random projections with a lag of k=4 frames; this signal was binarized using a threshold h=0.15 mm, summed across each of D=300 random projection dimensions, and then the resulting one-dimensional (1D) signal was smoothed with a Gaussian filter with a standard deviation of sigma=0.43 frames. Changepoints were identified as the local maxima of this smoothed 1D time-series. As specified above, this procedure depends in part upon the values of the parameters k, h and sigma; these were identified as those values that maximize the number of changepoints in the behaving mouse while yielding no change points in a dead mouse. Optimizing the difference between live and dead mice was largely insensitive to k, but sensitive to h and sigma; however, grid searching revealed clear optimal values for h and sigma, which are

used in the analysis above (Fig. S7B). Note that changepoints methods provide access to the overall timescale of behavior, but the specific boundaries between blocks identified by these approaches are highly approximate, as they only consider local data structure and ignore higher-order interactions between modules (including transition structure).

To ask whether any reasonably long snippet of behavior (greater than just a few frames) was ever "repeated" (without reliance on a underlying model for behavior), a template matching procedure was devised to identify similar trajectories through PCA space; here similarity was defined both quantitatively (minimal Euclidean distance in pose PC space) and qualitatively (via inspection of spines and PC trajectories). To identify similar trajectories, the Euclidean distance between some target snippet, the "template", and every possible snippet of equal length was computed. The most similar snippets were selected, ignoring snippets discovered that were shifted less than 1 second from each other (to ensure selection of behavioral snippets that occur distanced in time from each other).

Data Modeling and Fitting Methods

The modeling approach used herein takes preprocessed depth-camera video data (e.g., the 10 top PCs) as input and yields a fit model as output (Fig. S4A), where the fit model includes estimates of several key parameters, including the number of behavioral modules observed within a given set of experimental data, the autoregressive parameters that describe the motion expressed by the mouse within a given module, and the transition parameters that describe how often any particular module precedes or follows any other module; furthermore, for each video frame an assignment is made to the most likely associated behavioral module. To model pre-processed data, we modified established and well-documented modeling and fitting approaches

that have provided insight into complex and dense datasets with time-series structure (Fox et al., 2008; Fox et al., 2009). These approaches include Bayesian nonparametric methods, which allow identification of the most likely number of behavioral modules and lags in the AR process without human supervision. For convenience and clarity, we provide a summary of these methods and their specific application to the 3D behavioral data below, together with formal mathematical descriptions of our modeling and fitting procedures.

In brief, three-dimensional pose data were modeled using generative probabilistic modeling, which is often used to model complex dynamical processes (Bishop, 2006; Koller and Friedman, 2009; Murphy, 2012). By choosing probability models that reflect stylized versions of the physical dynamics that might give rise to the 3D pose dynamics data within a given experiment, this framework allowed the instantiation and testing of hypotheses about different organizations of behavior. These hypotheses ranged from simple progressions of poses described as Gaussians in pose space to the hierarchical AR-HMM that includes both autoregressive states and switching dynamics. Furthermore, using a generative framework for modeling behavior addresses a key limitation in alternative methods used for behavioral classification: distinct models for behavior can now be quantitatively and objectively compared based upon each model's ability to describe the statistics of held-out test data.

To test the hypothesis that behavior is composed of a series of behavioral modules that are stereotyped in form and which are interconnected in time with defined transition statistics, a family of discrete-time hidden Markov models (HMMs) were instantiated (Bishop, 2006; Koller and Friedman, 2009; Murphy, 2012). Here, at each point in time (e.g., for every frame of imaging data), an HMM posits that the mouse is within a discrete state that can be given a label. Each hidden Markov state corresponds to a prototypical brief three-dimensional motif of motion

the animal undertakes while within that state. The Markovian state of the dynamical process is then composed of both the latent discrete component, which identifies the behavioral mode of the animal, and a number of lagged values of the observation sequence, which are used by the autoregressive model to predict the short-timescale behavior of the animal based on the behavioral mode. This model structure is often called a switching vector-autoregressive (SVAR) model or autoregressive HMM (AR-HMM) (Fox et al., 2009; Murphy, 2012). Different experimental conditions were allowed to share the same library of state-specific VAR dynamics but learned their own transition patterns as well as any unique VAR dynamical modes, allowing the model to reveal changes in the parameters due to changes in the experiment.

Compositionally altering model structure allowed objective comparisons between different underlying models for how mouse behavior is organized over time. Removing the discrete switching dynamics captured in the transition matrix and replacing them with a mixture model generated an alternative model in which the distribution over each discrete state does not depend on its previous state. This would be the case if animals had a set of behavioral modules from which to choose, and the likelihoods of expressing any given one of them did not depend on the order in which they appear. This simplification resulted in the autoregressive mixture model (AR-MM). Alternatively, replacing the conditionally autoregressive dynamics with simple state-specific Gaussian emissions resulted in a Gaussian-emission HMM (G-HMM); this model explored the hypothesis that each behavioral module is best described by a simple pose, rather than being a dynamical trajectory. Applying both simplifications yielded a Gaussian mixture model (G-MM), in which behavior is simply a sequence of poses over time in which the probability of expressing any given pose does not depend on the prior pose. Removing the switching dynamics yielded a pure autoregressive (AR) or linear dynamical system (LDS)

model, in which behavior was described as a trajectory through pose space without any reused discrete behavioral modules at all. By fitting each of these alternative models and comparing model performance, the model structure that best explains the behavioral dynamics captured by the 3D imaging was identified (in this case the parent AR-HMM); this model (of the alternatives) thus best reflected the underlying organization of action (Fig. S4).

To fit models we utilized algorithms that have been well validated in a variety of contexts, including specifically for fitting models of the class used herein (Bishop, 2006; Koller and Friedman, 2009; Murphy, 2012 Robert, 2013, r14413). In brief, to estimate the parameters of any given model, approximate Bayesian inference was performed using Gibbs sampling, a Markov Chain Monte Carlo (MCMC) inference algorithm (Robert and Casella, 2013). The Gibbs sampling algorithm employed here has a natural alternating structure (Fig. S4C), directly analogous to the alternating structure of the popular expectation-maximization (EM) algorithm. Applied to the AR-HMM, the algorithm first segments the imaging data into modules given a fixed set of transition statistics and a fixed description of the AR parameters that describe any given module, and then the algorithm switches to fixing the segmentation and updating the transition matrix and the AR parameters. These two steps are alternated as an iterative process.

To employ Bayesian inference methods to fit the model, unknown quantities, including the transition matrix and the autoregressive parameters that describe each state, were treated with a uniform representation as latent random variables. In particular, weak prior distributions (except for the sticky parameter) were placed on these quantities and their posterior distributions after conditioning on observed 3D imaging data were investigated. For the autoregressive parameters, we used Automatic Relevance Detection (ARD) algorithms, which incorporate a prior with a Lasso-like penalty to encourage uninformative lag indices to have their

corresponding regression matrix coefficients tend to zero (Gelman et al., 2003). For the transition matrix, a Hierarchical Dirichlet Process (HDP) prior was used, which served to regularize the number of discrete latent states. In addition, the transition matrix prior also included a sticky bias, which is a single nonnegative number that controlled the tendency of the discrete states to self-transition (Fox et al., 2008; Fox et al., 2009). Because this parameter controlled the timescale of the inferred switching dynamics, this parameter was set such that the output of the model inference algorithms matches (as closely as possible) the model-free duration distribution determined by our changepoint analysis. In this sense the sticky bias acts as a kind of "lens" which focuses the model on the organization of behavior at a specific temporal scale. It is important to note that despite the fact that this parameter was tuned — indeed this parameter defined our prior over the timescale of behavior — it was not clear *a priori* that the model could identify a duration distribution that was similar to the changepoints distribution (or that a duration distribution mode would match the specific sub-second timescale found in the model-free analysis) for any sticky parameter value.

Model Training and Parameter Tests

Datasets from the open-field, odor, and genetic manipulation experiments were modeled jointly in a single model to increase statistical power and facilitate comparisons between results. Because the neural implants associated with the optogenetics experiment modestly altered the physical profile of the animal, these data were modeled separately. In all experiments, the first 10 principal components for the wavelet decomposition coefficients of each frame of each imaged mouse were gathered. Data were then randomly subdivided and assigned either a "train" or a "test" label, with 70% of the data assigned to the training subset, and 30% assigned to the test

subset. The mice labeled “test” were held-out from the training process, and used to test generalization performance via measurement held-out likelihood. This approach allowed us to directly compare algorithms whose composition reflected different underlying structures for behavior.

Models were trained on data using the procedures described above; modeling was robust to both initialization settings and to parameter and hyperparameter settings (with the exception of the sticky bias parameter kappa, see below). Specifically, the number of lags used in our AR observation distributions and the number of used states in our transition matrix with an HDP prior was found to be robust to the particular hyperparameter settings on both. The hyperparameters of our sparsifying ARD prior were varied by several orders of magnitude, and held-out likelihood, the number of used lags, and the number of used states varied negligibly. The hyperparameters of our HDP prior were also varied by several orders of magnitude and again no change to the number of used states or held-out likelihood was observed. All jointly-trained data shared observation distributions, but each treatment class was allowed its own transition matrix. Each model was updated through 1000 iterations of Gibbs sampling; upon the last iteration of Gibbs sampling the model output was saved; all further analysis was performed on this final update.

The “stickiness” of the duration distribution of our behavioral modules — defined by the kappa setting of the model — influenced the average duration of behavioral modules discovered by the AR-HMM; this allowed us to control the temporal scale at which behavior was modeled. As discussed in the main text, autocorrelation, power spectral density, and the changepoint algorithm identified switching dynamics at a specific sub-second temporal scale (as encapsulated by the changepoints duration distribution and reflected by the spectrogram and autocorrelogram).

The kappa stickiness parameter of the time-series model was therefore empirically set to best match the duration distribution discovered by changepoint detection. To find the kappa setting at which these distributions were best matched, the Kolmogorov-Smirnov distance was minimized between the inter-changepoint interval distribution and the posterior behavioral module duration distribution through a dense grid search. Using alternative distance metrics did not affect the results.

Evaluating the Log-likelihood of Held-out Test Data

A crucial feature of a useful model is its ability to "generalize", meaning its ability to usefully describe phenomena in similar data that it has not previously seen. This is intimately related to the problem of over-fitting; if a model performs well on training data, but poorly on some new data, then the model has been over-fit. The higher the generalization performance, which was measured as the log-likelihood of held-out test data, the less the model has been over-fit. Model comparisons were performed by evaluating the log-likelihood of held-out test data under the AR-HMM and each member of the family of alternative models. For each Markov Chain Monte Carlo (MCMC) sample of the AR-HMM transition matrix and autoregressive parameters, the log likelihood of a held-out data sequence was computed by summing over every possible hidden state sequence the joint probability of the data sequence and that hidden state sequence (Bishop, 2006).

Predictive comparison metric to stratify model capacities

We use two metrics to compare model performance: the aggregate held-out likelihood score used to compare models in Figure S4D, and a temporal horizon-based predictive

comparison metric in Figure 2A. While the held-out likelihood score used in Fig. S4D compares the models' overall prediction performance, the metric shown in Figure 2A shows separately the benefit of each component of the model's structure.

There are two possible versions of the horizon-based prediction metric, both based on predictive likelihood but differing in what data is used to make the prediction:

1. Given the data up to time t, score how well the model predicts what happens at time t+k (for various k>0 and averaging over t). Symbolically, this metric is

$$\log p(y_{t+k}|y_1, \ldots, y\_t)$$

2. Given the data up to time t *and an instant of data before time t+k for the short timescale model*, score how well the model predicts what happens at time t+k. More precisely, an "instant of data" is taken to be the amount of data used in the short-timescale AR model. Symbolically, this metric for the AR-HMM, AR-MM, and AR models is

$$\log p(y_{t+k}|y_1, \ldots, y_t, y_{t+k-3}, y_{t+k-2}, y_{t+k-3})$$

where the short-timescale data is not included for the G-HMM, G-MM, or Gaussian models (i.e. the "pose" models).

Metric #1 shows overall prediction capability, much like the comparison in Figure S4D, and hence lumps together the benefit from the AR model component and that from the discrete hidden state model component. However, Metric #2 actually separates these effects out: the amount by which the AR-HMM beats the AR-MM is precisely due to the discrete Markov state, since both models use the same short-timescale information. We therefore use Metric #2 for Fig. 2. Metric #2 demonstrates that the AR-MM persistently beats the AR model, corresponding to the fact that it is a more expressive dynamics model for short timescales. Metric #2 also shows

the G-HMM and G-MM (i.e. the "pose models") as persistently lower because they do not model the short timescale information. Finally, we note that Metric #2 can be computed exactly, while #1 requires approximation with Monte Carlo or importance sampling methods.

Analysis of the Model Consistency

To evaluate model consistency the following procedure was adopted, which enabled comparisons between model fits. For any two fits, each containing a set of behavioral modules, the modules between the two fits were matched by the Frobenius norm between the AR matrices for each module, and then the Hungarian algorithm (also called Munkres matching) was used to generate a mapping that associated a module in the first fit with a single module in the second fit. The number of frames each fit assigned to the module was then plotted as a single point, where the x-axis represents frames assigned to that module in the first fit, and the y-axis represents frames assigned to that module in the second fit. We evaluate how well two fits matched as the goodness-of-fit R^2 value between these points and the y=x line. Both algorithmic consistency, in which the data do not change, but the random seed that initializes the model is different, and data consistency, in which different bootstrapped datasets are used to train the model, were evaluated with this procedure. Results for algorithmic consistency are described in the main text ($R^2 = 0.93 \pm 0.03$). To test data consistency, we randomly sampled a subset of n=24, 20, 10 and 5 mice from a dataset with 25 mice (20 minutes each in the OFA), and refit the AR-HMM model 15 times. For n=24: $R^2 = 0.95 \pm 0.02$; n=20: $R^2 = 0.93 \pm 0.04$; n=10: $R^2 = 0.92 \pm 0.04$; n=5: $R^2 = 0.88 \pm 0.09$.

Analysis on Model Output

Fit models included a number of states that were infrequently used (see Fig. S5E). Because we wished to focus on those changes in behavior that are most significant, we concentrated our analysis on the set of modules that explain 95 percent of the total frames in our dataset (e.g., the top 51 modules in the dataset) and in the main text only describe modules within this group. Visual inspection reveals that many of the frames assigned to states above this threshold are corrupted by sensor-specific noise that is not eliminated by our pre-processing. Because the model attempts to cluster together data with similar structure, this thresholding procedure has a practical de-noising effect and mitigates ambiguity injected into the imaging by the depth camera itself. Future use of our approach with different depth cameras may require distinct pre-processing and thresholding methods, as alternative cameras will have unique noise characteristics.

To represent the grammatical relationship between behavioral syllables, two analyses were performed. First, the probabilities of transition from any given module to any other module (in a pairwise fashion) were computed; second, the bigram probability was computed, which is the probability that two syllables were found occurring one after the other (a "bigram" of modules), as a fraction of all observed bigrams. To calculate this value for each pair (i,j) of modules, first a square n x n matrix, A, was created where n is the number of total modules in the label sequence. The label sequences that were saved at the last iteration of Gibbs sampling were then scanned, incrementing the entry A[i,j] for every time syllable i directly preceded syllable j. At the end of the label sequence, each entry was divided by the number of total bigrams observed.

In order to visually organize those modules that were specifically up-regulated or selectively expressed as a result of a manipulation, a selectivity index was assigned to each

module. Specifically, where p(condition) indicates the percent usage of a module in a condition, modules in the circular open field versus square box comparison were sorted by (p(circle)-p(square) / (p(circle) + p(square)). In the comparison between blank odor and fox odor (TMT), modules were sorted by (p(tmt) - p(blank)) / (p(tmt) + p(blank)). In the comparison of *Ror1β* mutant, heterozygous mutant, wildtype and C57/BL6, modules were sorted by (p(het)+p(mut)) / (p(wt) + p(c57)).

Statemap Visualizations

To lay out each graph in a reproducible manner (up to global rotation of the figure), the position of the nodes was initialized using the spectral layout algorithm and fine-tuned node positions using the Fructherman-Reingold iterative force-directed layout algorithm (Koren, 2005); both algorithms were used as implemented in the NetworkX and GraphViz software package. To simplify the visual representation, only edges for which the bigram value was more than some multiple of bootstrap-estimated standard error units (indicated in figure legends) above zero were drawn.

Hinton Diagram

To succinctly display all bigram probabilities between all syllables, the full bigram probability matrix was displayed using a Hinton diagram. In this representation, the value at each index of a rectangular matrix is shown as a 2D geometric object (such as a square or circle) at the spatial index of the matrix's element, where the area of the geometric element is proportional to the value of the matrix at that index. This has the property of representing larger values as more visually salient than smaller values. Additionally, we encoded additional properties of the matrix,

such as the statistical significance of a particular bigram, using color of each geometric element. Because the indices of syllables in our AR-HMM model are chosen randomly by our model, we resorted the rows and columns of the bigram probability matrix using spectral clustering to better represent syllables which tend to transition amongst themselves. This re-sorting did not affect the data itself, only its presentation.

Computing Entropy Rate and Mutual Information

Let A denote the estimated transition matrix, so that $A_{ij} = \mathrm{P}[x_{t+1} = j \mid x_t = i]$ for $i, j = 1,2, \ldots, n$ and all time indices t, and let π denote the steady-state distribution of the Markov chain, so that $\pi_i = \Pr[x_t = i]$. The entropy rate on the transition matrix was calculated as $-\sum_{i,j} \pi_i A_{ij} \ln A_{ij}$. The mutual information between states was calculated as $-\sum_i \pi_i \ln \pi_i + \sum_{i,j} \pi_i A_{ij} \ln A_{ij}$.

Cross-likelihood Analysis of AR-HMM States

To evaluate whether discovered AR-HMM states represent meaningful clusters of data in the underlying dataset, we measured how well, on average, each state's autoregressive parameters explain its assigned data segments compared to how well, on average, they explain data segments assigned to other states. This cross-likelihood measure can also be viewed in terms of likelihood ratio testing: given a data segment to assign to one of two possible states, if that data segment was assigned according to a likelihood ratio test, the corresponding cross-likelihood values measure how far on average such data segments are from the likelihood ratio test decision boundary. Cross-likelihoods near or above 1 (i.e., when their logarithms are near or above 0) indicate that the corresponding states' autoregressive models can explain each others'

data segments; the states' assigned data segments cannot be readily or reliably discriminated, at least using the states' autoregressive parameters and a likelihood ratio test. Conversely, likelihoods well below 1 indicate that the states' assigned data segments are easily discriminated.

For any two states $i, j \in \{1,2, \ldots, n\}$ with assigned data segments $\{y_k^{(i)}\}_{k=1}^{N_i} = \{y_{t_a:t_b} : x_{t_a:t_b} = i, x_{t_a-1} \neq i \neq x_{t_b+1}\}$ and $\{y_k^{(j)}\}_{k=1}^{N_j} = \{y_{t_a:t_b} : x_{t_a:t_b} = j, x_{t_a-1} \neq j \neq x_{t_b+1}\}$ and autoregressive parameters $\theta^{(i)}$ and $\theta^{(j)}$, respectively, we computed the (i, j) entry of the log cross-likelihood matrix as

$$\mathrm{CL}_{ij} \stackrel{\text{def}}{=} \frac{1}{N_j} \sum_{k=1}^{N_j} \log \frac{p\left(y_k^{(j)} \mid \theta^{(i)}\right)}{p\left(y_k^{(j)} \mid \theta^{(j)}\right)}$$

While these cross-likelihood ratios can be interpreted on an absolute scale in terms of classification of data segments using likelihood ratio testing, it is also useful to compare this measure to a null model (i.e., a reasonable baseline hypothesis that should not have meaningful discrete state structure). This baseline was constructed by first fitting with Gibbs sampling a linear dynamical system (LDS) with 60-dimensional latent state to the same open field dataset on which AR-HMM was fit; while an LDS can capture the correlation structure in the data, both in time and across components, its only state is continuous and thus it by definition lacks modularity. Synthetic data were generated from the fit LDS; an AR-HMM was then fit to the synthetic data, thereby producing the corresponding cross-likelihood plot from the inferred states. In this specific case the AR-HMM hyperparameters were adjusted to enable more than one state to be inferred; otherwise, the Bayesian inference procedure only instantiates one autoregressive state to explain the LDS synthetic data, a result that is consistent with our non-parametric methods appropriately recognizing state number in the underlying data.

Visualizing Synthetic Mice

To generate synthetic mice from the AR-HMM and each model in the family of alternative models, the sticky kappa biasing term in each model was replaced with an explicit duration model so as to more accurately represent the timescales in each model's inferred label sequence. For each model fit, a corresponding autoregressive hidden semi-Markov model (AR-HSMM) was instantiated with the same autoregressive and transition parameters but with a negative binomial duration model for each state, which was also fit using Gibbs sampling. Each AR-HSMM generates a 10D principal coordinate sequence; using the principal components analysis fit to the original data, these are mapped into images to produce the "dream mouse" videos.

Statistical Methods

To evaluate the model's performance from a frequentist perspective, the inference algorithm is treated as a (stochastic) point estimation procedure, and how that point estimate might change if the estimation procedure were applied to a new random dataset was explored. In particular, a bootstrap procedure was used to estimate the distribution of these point estimates under new random datasets (see below). In addition, tools from hypothesis testing were used to summarize the degree to which the measured module usage differences could be explained by random fluctuation or finite-sample effects. To evaluate the model's performance from a Bayesian perspective, the posterior uncertainty estimated by our MCMC inference algorithm was reported in terms of both raw samples and coordinate-wise posterior standard deviation error bars.

For the purposes of evaluating the procedure as an estimator that might be applied to new random datasets, the last MCMC sample was treated as a (stochastic) point estimate. To estimate the distribution of that point estimate under new random datasets, a bootstrap was performed: the underlying population distributions over sequences for each group (e.g., blank, TMT) were approximated by their empirical distributions, and sampling with replacement from those distributions generated new random datasets. Under the assumption that the data are generated from a stationary random process, this bootstrap procedure provides a consistent estimate of the distribution of the quantities estimated by the model inference, one that has been well validated in a wide range of contexts (Wasserman, 2013).

To solve the label matching problem in the bootstrap samples, the new fits were initialized from the first fit on the full (un-bootstrapped) dataset; this way the variations in the bootstrap fits reflected the variations due to new random datasets and not any label matching effects (see the model consistency section above for methods that enable independent evaluation of label matching in fits with distinct random initializations). Figures 3-5 show the bootstrap samples and resulting standard error estimates for the module usages in each experiment. In the main text, all additive statistical errors and error bars are bootstrap-estimated standard errors (±SE) unless otherwise indicated. We also computed Bayesian uncertainty estimates from MCMC samples. In Fig. S5E we show the last 1000 sampled module usage vectors in a 1250 iteration run of our MCMC algorithm. The small variation indicates that the posterior mode being explored by the MCMC algorithm is highly peaked.

To summarize the degree to which the differences in module usage vector estimates might be explained by random fluctuation and finite sample sizes, null hypotheses were formulated in terms of asymptotic normality assumptions on the distributions of the estimates,

which was validated using Probability-Probability plots (data not shown). To define a single overall null hypothesis that the usage vectors of two populations are the same up to statistical variation, let u and v denote vectors of estimated usage proportions for two populations (e.g., blank and TMT). We test the null hypothesis that u - v is distributed according to a multivariate normal distribution

$$\mathrm{H}_0 : \; u - v \backsim \mathrm{Normal}(0, \Sigma)$$

where Σ is estimated from the bootstrap samples. We used a standard test statistic, Hotelling's T-squared statistics, for H_0.

In addition to testing overall hypotheses about whether module usage vectors are the same in two populations, we also formulated p-values for hypotheses on the individual module usages to bound the (estimated) Type I family-wise error rate. To define these families of module-wise null distributions and p-values, asymptotic normality assumptions were again employed: letting u and v denote the estimated module usage vectors for two populations and letting u_i and v_i denote their ith coordinates, respectively, for each module i we define the hypothesis

$$\mathrm{H}_0^{(i)} : \; u_i - v_i \sim \mathrm{Normal}(0, \sigma_i^2)$$

where the σ_i^2 are estimated from the bootstrap samples. For each such hypothesis an un-adjusted p-value was calculated using a simple Wald test. To adjust the p-values, the Holm-Bonferroni step-down procedure was used, which is a slightly less conservative variant of the Bonferroni correction with the same family-wise error rate guarantees.

The p-values in the optogenetic stimulation experiment were computed differently from the statistics described above. Let $u_i^{(t)}$ denote the frequency (across the 50 trials for a single fit) with which module i is used at frame index t relative to stimulus onset for $t \in [-1, 6]$ seconds

(with 30 frame indices per second). We modeled each $u_i^{(t)}$ as a random variable with a distribution that may depend on the stimulus; that is, we partitioned the time relative to stimulus onset into three subsets corresponding to times when the laser is on ($0 \leq t < 2$), times just after the laser turns off ($2 \leq t < 3$), and all other times. The family of hypotheses that

$$\mathrm{H}_0^{(\text{i, on vs off})} : \; u_i^{(t_1)} - u_i^{(t_2)} \sim \mathrm{Normal}\left(0, \sigma^2_{i,\text{ on vs off}}\right) \quad 0 \leq t_1 < 2, t_2 \in [-1,0) \cup (3,6]$$

$$\mathrm{H}_0^{(\text{i, offset vs off})} : \; u_i^{(t_1)} - u_i^{(t_2)} \sim \mathrm{Normal}\left(0, \sigma^2_{i,\text{ offset vs off}}\right) \quad 2 \leq t_1 < 3, t_2 \in [-1,0) \cup (3,6]$$

was then tested, where each null distribution's variance was estimated from the sample variance across time indices. P-values were then computed using a t-test and adjusted with the Holm-Bonferroni step-down procedure.

Technical definitions of the generative models

In this section, the priors and generative models implemented in this paper are formally defined following standard notation. These generative models are adaptations of models successfully used previously to characterize structure in complex datasets (Fox et al., 2009; Murphy, 2012).

Prior on the transition matrix

A sticky HDP prior was placed on the transition matrix π with concentration parameters $\alpha, \gamma > 0$ and sticky parameter $\kappa > 0$

$$\rho_j \overset{\text{iid}}{\sim} \mathrm{Beta}(1, \gamma) \qquad \beta_i = (1 - \rho_i) \prod_{j<i} \rho_j$$

$$\pi_i \overset{\text{iid}}{\sim} \mathrm{DP}(\alpha\beta + \kappa\delta_i) \qquad i = 1,2,\ldots$$

where δ_{ij} is 1 when $i = j$ and is 0 otherwise and π_i denotes the ith row of π. Gamma priors are placed on α and γ, setting $\alpha \sim \text{Gamma}(1,1/100)$ and $\gamma \sim \text{Gamma}(1,1/100)$.

Generation of the discrete state sequence

Given the transition matrix, the prior on a discrete state sequence x was

$$x_t \sim \pi_{x_{t-1}} \quad t = 2,3,\ldots,T$$

where x_1 is generated by the stationary distribution under π.

Prior on the autoregressive parameters

The autoregressive parameters $\theta = \{\theta^{(i)}\}_{i=1}^{\infty} = \{A^{(i)}, b^{(i)}, \Sigma^{(i)}\}$ for each state $i = 1,2,\ldots$ were sampled from a Matrix Normal Inverse-Wishart prior :

$$(A, b), \Sigma \sim \text{MNIW}(\nu_0, S_0, M_0, K_0)$$

or equivalently

$$\begin{aligned} \Sigma &\sim \text{InvWishart}(\nu_0, S_0) \\ \text{vec}((A, b)) &\sim \text{Normal}(\text{vec}(M_0), \Sigma \otimes K_0) \end{aligned}$$

where $\otimes$ denotes a Kronecker product and (A, b) denotes the matrix formed by appending b to A as a column. Note that the noise covariance matrices $\Sigma^{(i)}$ are dense full-rank matrices, not restricted to diagonal matrices (i.e. uncorrelated noise models), and there is more than enough data to learn these fully general noise models. In addition, a block ARD prior(Fox et al., 2009; Pole et al., 1994) on K_0 is used to encourage uninformative lags to be shrunk to zero:

$$K_0 = \text{diag}(k_1, \ldots, k_{KD}) \qquad k_i \overset{\text{iid}}{\sim} \text{InvGamma}(1/25,1/25)$$

Generation of the 3D pose sequence principal components

Given the autoregressive parameters and discrete state sequence, the data sequence y was generated according to an affine autoregression :

$$y_t \sim \text{Normal}(A^{(x_t)}\tilde{y}_{t-1} + b^{(x_t)}, \Sigma^{(x_t)}) \quad t = K+1, K+2, \ldots, T$$

where the covariance matrices Σ are dense and $\tilde{y}$ denotes a vector of K lags:

$$\tilde{y}_t \overset{\text{def}}{=} [y_{t-K}^T \quad y_{t-K+1}^T \quad \cdots \quad y_{t-1}^T]^T$$

The alternative models are special cases of the AR-HMM and were constructed by adding constraints. In particular, the Gaussian-emission HMM (G-HMM) corresponds to constraining $A^{(i)} = 0$ for each state index i. Similarly, the autoregressive mixture (AR-MM) and Gaussian mixture (GMM) correspond to constraining the transition matrix to be constant across rows, $\pi_{ij} = \pi_{i'j} = \pi_j$ for each i and i', in the AR-HMM and G-HMM, respectively.

Technical description of the inference algorithms

As discussed above, the Gibbs sampling inference algorithm alternated between two principal stages: updating the segmentation of the data into modules given a fixed transition matrix and autoregressive parameters, and updating the transition matrix and autoregressive parameters given a fixed segmentation. These fitting procedures are adaptations of those successfully used previously to fit generative models including the AR-HMM (Fox et al., 2008; Fox et al., 2009). Mathematically, updating the segmentation sampled the label sequence x conditioned on the values of the data y, the autoregressive parameters θ, and the transition matrix π; that is, sampling the conditional random variable $x \mid \theta, \pi, y$. Similarly, updating the transition matrix and autoregressive parameters given the segmentation sampled $\pi \mid x$ and $\theta \mid x, y$, respectively.

For inference in the AR-HMM the weak limit approximation to the Dirichlet process was used, in which the infinite model was approximated by a finite one. That is, choosing some finite approximation parameter L, β and π were modeled using finite Dirichlet distributions of size L

$$\beta \sim \mathrm{Dir}(\gamma/\mathrm{L}, \cdots, \gamma/\mathrm{L})$$

$$\pi_k \sim \mathrm{Dir}(\alpha\beta_1, \cdots, \alpha\beta_j + \kappa\delta_{kj}, \cdots, \alpha\beta_L)$$

where π_k denotes the ith row of the transition matrix. This finite representation of the transition matrix allowed the state sequence x to be resampled as a block and for large L provides an arbitrarily good approximation to the infinite Dirichlet process.

Using a weak limit approximation, the Gibbs sampler for the AR-HMM iterated resampling the conditional random variables

$$x \mid \pi, \theta, y \quad \theta | x, y \quad \text{and} \quad \beta, \pi | x$$

For simplicity, throughout this section notation for conditioning on hyperparameters and the superscript notation for multiple observation sequences is suppressed.

Sampling $x \mid \pi, \theta, y$

Sampling the state labels x given the dynamical parameters, π and θ, and the data y corresponds to segmenting the 3D video sequence and assigning each segment to a behavioral mode that describes its statistics.

Given the observation parameters θ and the transition parameters π, the hidden state sequence x is Markov with respect to a chain graph. The standard HMM backward message passing recursions are

$$B_t(k) = p(y_{t+1:T} | \theta, \pi, x_t = k)$$

$$= \sum_{j=1}^{K} p(x_{t+1} = j \mid x_t = k, \pi)\, p(y_{t+1} \mid x_{t+1} = j, \theta)\, B_{t+1}(j)$$

for $t = 1,2, \ldots, T-1$ and $k = 1,2, \ldots, K$, where $B_T(k) = 1$ and where $y_{t+1:T} = (y_{t+1}, y_{t+2}, \ldots, y_T)$. Using these messages, the conditional distribution of the first state x_1, marginalizing over all the future states $x_{2:T}$ is

$$p(x_1 = k \mid \pi, \theta, y) \propto p(x_1 = k \mid \pi) p(y_1 \mid x_1 = k, \theta) B_1(k)$$

which can be sampled efficiently. Given a sampled value $\bar{z}_1$, the conditional distribution of the second state x_2 is

$$p(x_2 = k \mid \pi, \theta, y, x_1 = \bar{z}) \propto p(x_2 = k \mid x_1 = \bar{z_1}, \pi)\, p(y_2 \mid x_2 = k, \theta) B_2(k).$$

Therefore after passing HMM messages backward the state sequence can be recursively sampled forwards.

Sampling $\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y}$

Sampling the autoregressive parameters θ given the state sequence x and the data sequence y corresponds to updating each mode's dynamical parameters to describe the 3D video data segments assigned to it.

To resample the observation parameters θ conditioned on a fixed sample of the state sequence x and the observations y one can exploit conjugacy between the autoregressive likelihood and the MNIW prior. That is, the conditional also follows the MNIW distribution:

$$p\left(A^{(k)}, \Sigma^{(k)} \middle| x, y, S_0, \nu_0, M_0, K_0\right) = p\left(A^{(k)}, \Sigma^{(k)} \middle| S_n, \nu_n, M_n, K\_n\right)$$

where (S_n, ν_n, M_n, K_n) are posterior hyperparameters that are functions of the elements of y assigned to state k as well as the preceding lagged observations:

$$S_n = S_0 + S_{yy^\top} + (M_0 K_0^{-1} M_0^\top - M_n K_n^{-1} M_N^\top)$$

$$M_n = \left(M_0 K_0^{-1} + S_{\tilde{y}\tilde{y}^\top}\right) K_n$$

$$K_n = \left(K_0^{-1} + S_{\tilde{y}\tilde{y}^\top}\right)^{-1}$$

$$\nu_n = \nu_0 + n$$

where

$$S_{yy^\top} = \sum_{t:x_t=k} y_t\, y_t^\top \quad S_{\tilde{y}\tilde{y}^\top} = \sum_{t:x_t=k} \tilde{y}_t\, \tilde{y}_t^\top$$
$$S_{y\tilde{y}^\top} = \sum_{t:x_t=k} y_t\, \tilde{y}_t^\top \quad n = \#\{t: x_t = k\}.$$

Therefore resampling $\theta | \, x, y$ includes three steps: collecting statistics from the data assigned to each state, forming each state's posterior hyperparameters, and updating each state's observation parameter by simulating a draw from the appropriate MNIW. Simulating $(A, \Sigma) \sim \text{MNIW}(S_n, \nu_n, M_n, K_n)$ proceeds as

$$\Sigma \sim \text{InvWishart}(S_n, \nu_n)$$
$$A = M_n + \Sigma^{\frac{1}{2}}\, G\, \mathrm{K_n}^{-\frac{1}{2}} \quad \text{where} \quad G_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0,1).$$

Sampling $\boldsymbol{\beta}, \boldsymbol{\pi} | \, \boldsymbol{x}$

Sampling the transition parameters π and β given the state sequence x corresponds to updating the probabilities of transitions among behavioral modules to reflect the transition patterns observed in the state sequence. Updating β encouraged redundant behavioral modes to be pruned from the model, while updating each π_{ij} fit the transitions observed from state i to state j.

Resampling the transition parameters β and π, which are draws from the weak limit approximation to the (sticky) HDP, was performed using an auxiliary variable sampling scheme . That is, $\beta,\backslash pi \mid x$ was generated by first sampling auxiliary variables $m \mid \beta, x$. Then $\beta,\backslash pi \mid x, m$ was generated by first sampling from the marginal $\beta \mid m$ and then the conditional $\pi \mid \beta, x$.

The matrix of transition counts in the sampled state sequence x is

$$n_{kj} = \#\{t: x_t = k,\ x_{t+1} = j,\ t = 1,2, \dots, T-1\}.$$

Suppressing conditioning notation for simplicity, the auxiliary variables $m = \{m_{kj}: k, j = 1,2, \dots, K\}$ are sampled via

$$m_{kj} = \sum_{l=1}^{n_{kj}} b_{kjl} \quad \text{where} \quad b_{kjl} \overset{\text{iid}}{\sim} \text{Bernoulli}\left(\frac{\alpha\beta_j}{\alpha\beta_j + \kappa} \frac{\alpha\beta_j + \kappa\delta_{kj}}{\alpha\beta_j + l + \kappa\delta_{kj}}\right)$$

where $\text{Bernoulli}(p)$ denotes a Bernoulli random variable that takes value 1 with probability p and takes value 0 otherwise. Note that the update for the HDP-HMM without a sticky bias corresponds to setting $\kappa = 0$ in these updates.

Given the auxiliary variables, the update to β is a Dirichlet-multinomial conjugate one, where

$$\beta \mid m \sim \text{Dir}(\frac{\gamma}{K} + m_{\cdot 1}, \frac{\gamma}{K} + m_{\cdot 2}, \cdots, \gamma/K\ + m_{\cdot K})$$

where $m_{\cdot j} = \sum_{k=1}^{K} m_{kj}$ for $j = 1,2, \dots, K$. The update to $\pi \mid \beta, x$ is similar, with

$$\pi_k \mid \beta, x \sim \text{Dir}\big(\alpha\beta_1 + n_{k1}, \cdots, \alpha\beta_j + n_{kj} + \kappa\delta_{kj}, \cdots, \backslash alpha\ \beta_K + n_{kK}\big).$$

Supplemental References

Arias-Castro, E., and Donoho, D.L. (2007). Does Median Filtering Truly Preserve Edges Better Than Linear Filtering?

Basseville, M., and Nikiforov, I.V. (1993). Detection of abrupt changes (Prentice-Hall Publishing).

Bishop, C.M. (2006). Pattern Recognition and Machine Learning (Springer).

Fox, E.B., Sudderth, E.B., Jordan, M.I., and Willsky, A.S. (2008). An HDP-HMM for Systems with State Persistence. In Proc International Conference on Machine Learning.

Fox, E.B., Willsky, A.S., Sudderth, E.B., and Jordan, M.I. (2009). Sharing Features among Dynamical Systems with Beta Processes. Advances in Neural Information Processing Systems 22 (NIPS 2009), 549-557.

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. (2003). Bayesian Data Analysis, Second Edition (Chapman & Hall/CRC Texts in Statistical Science). (29 July 2003).

Koller, D., and Friedman, N. (2009). Probabilistic Graphical Models (MIT Press).

Koren, Y. (2005). Drawing graphs by eigenvectors: theory and practice. Computers & Mathematics with Applications *49*, 1867-1888.

Mallat, S.G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. Pattern Analysis and Machine Intelligence, IEEE Transactions on *11*, 674-693.

Murphy, K.P. (2012). Machine Learning (MIT Press).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V.*, et al.* (2011). Scikit-learn: Machine Learning in Python. The Journal of Machine Learning Research *12*, 2825-2830.

Perez, F., and Granger, B.E. (2007). IPython: a system for interactive scientific computing. Computing in Science & Engineering.

Pole, A., West, M., and Harrison, J. (1994). Applied Bayesian Forecasting and Time Series Analysis (CRC Press).

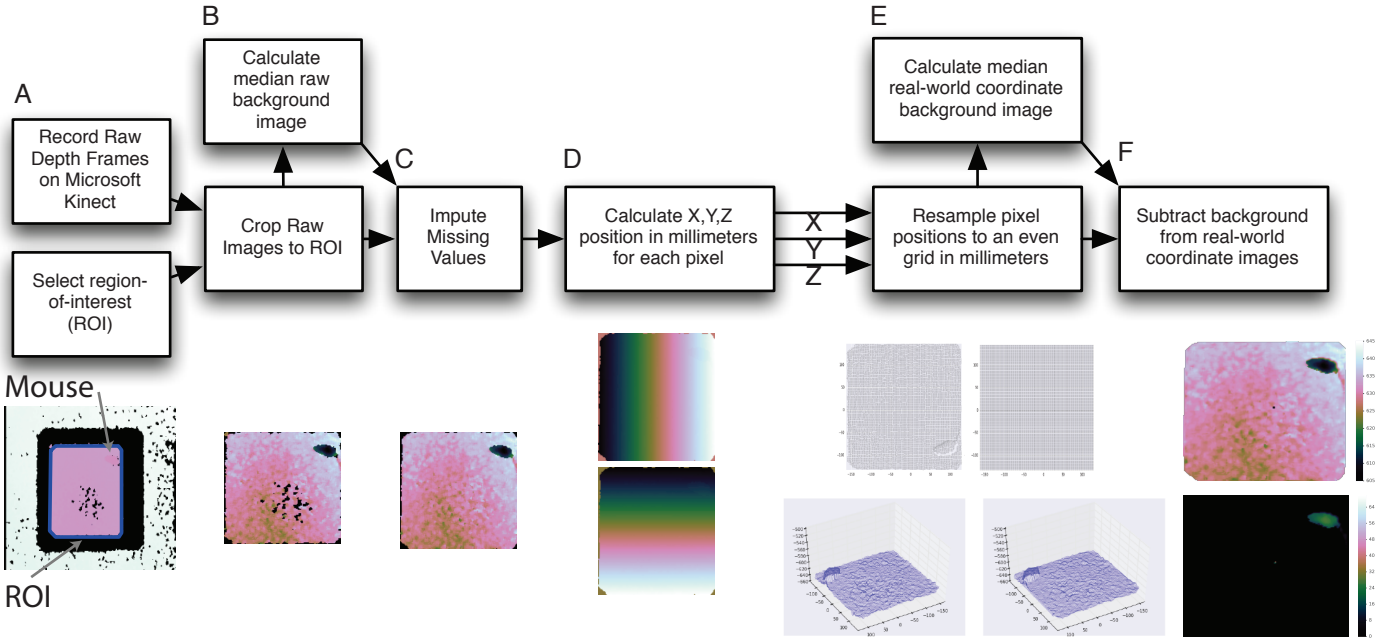
Robert, C., and Casella, G. (2013). Monte Carlo Statistical Methods (Springer Science & Business Media).

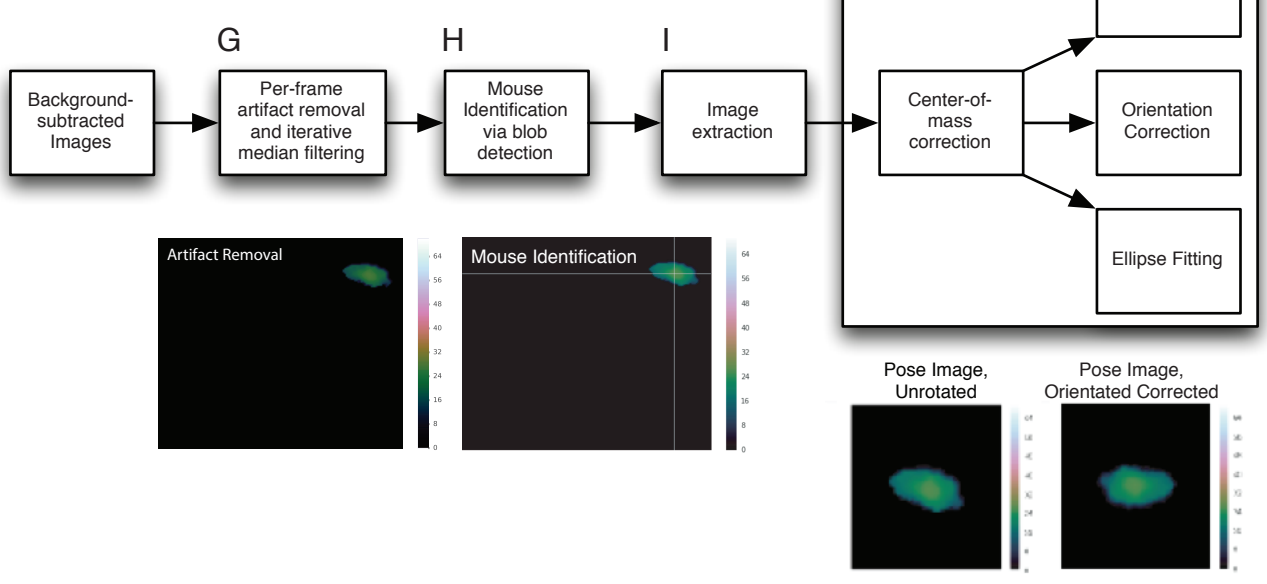
van der Walt, S., Colbert, S.C., and Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering *13*, 22-30.

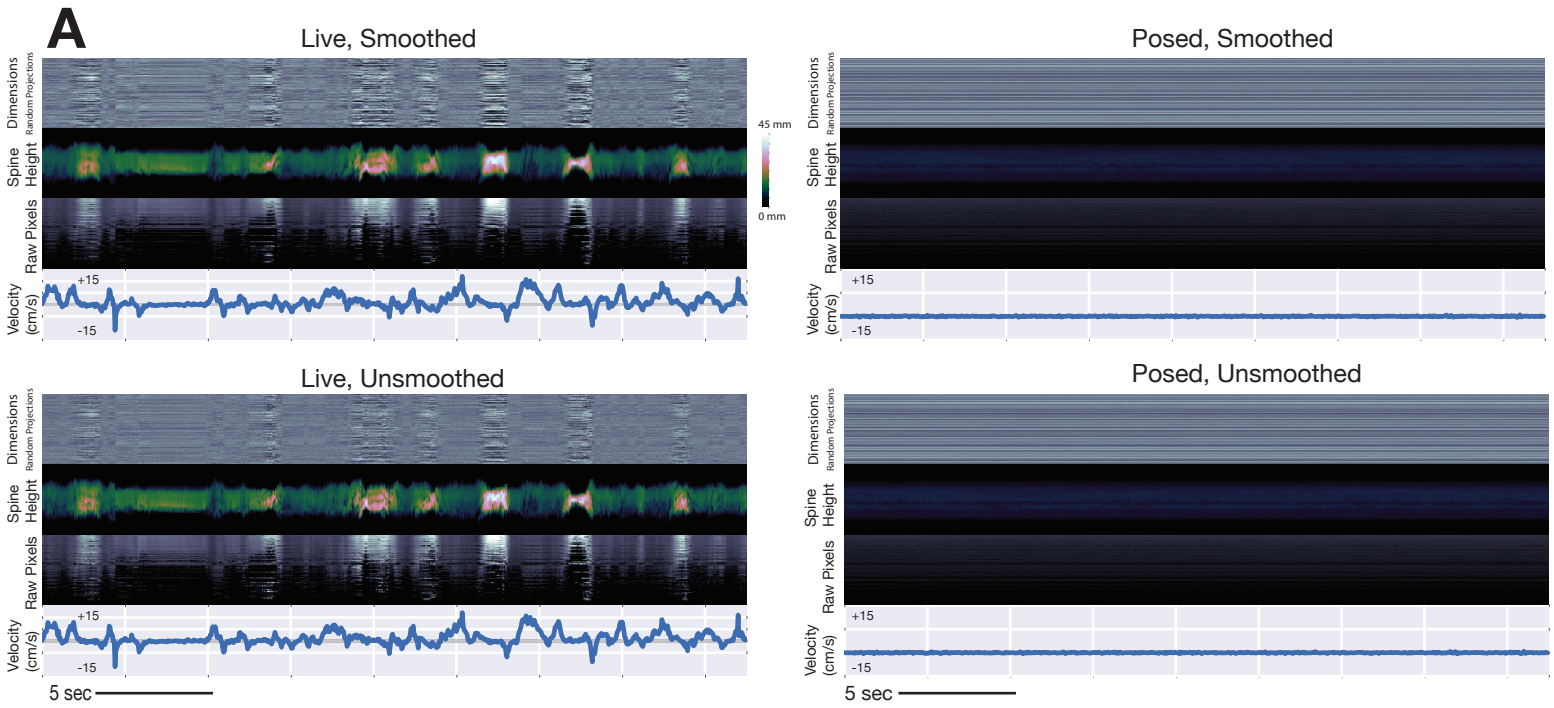
Wasserman, L. (2013). All of Statistics (Springer Science & Business Media).

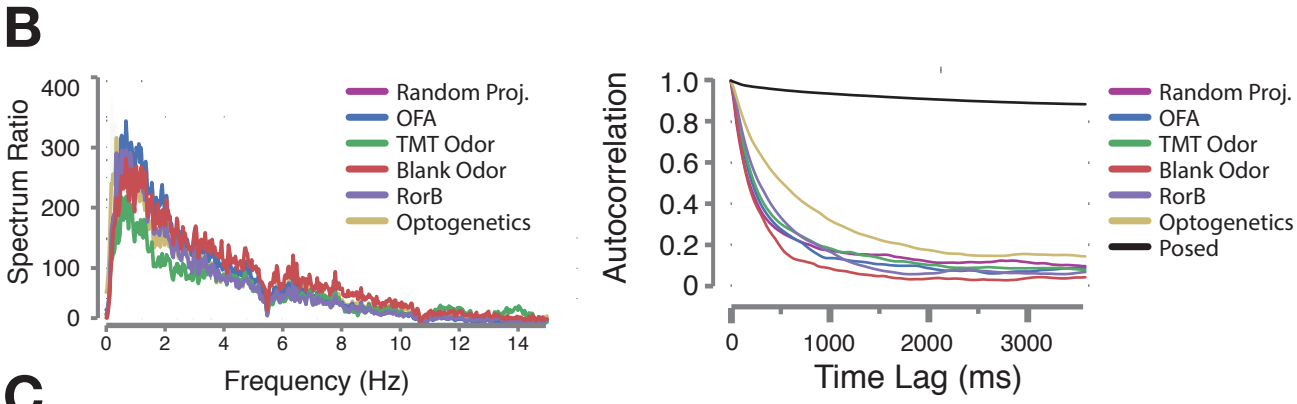
Welch, P. (1967). The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. IEEE Transactions on Audio and Electroacoustics *15*, 70-73.

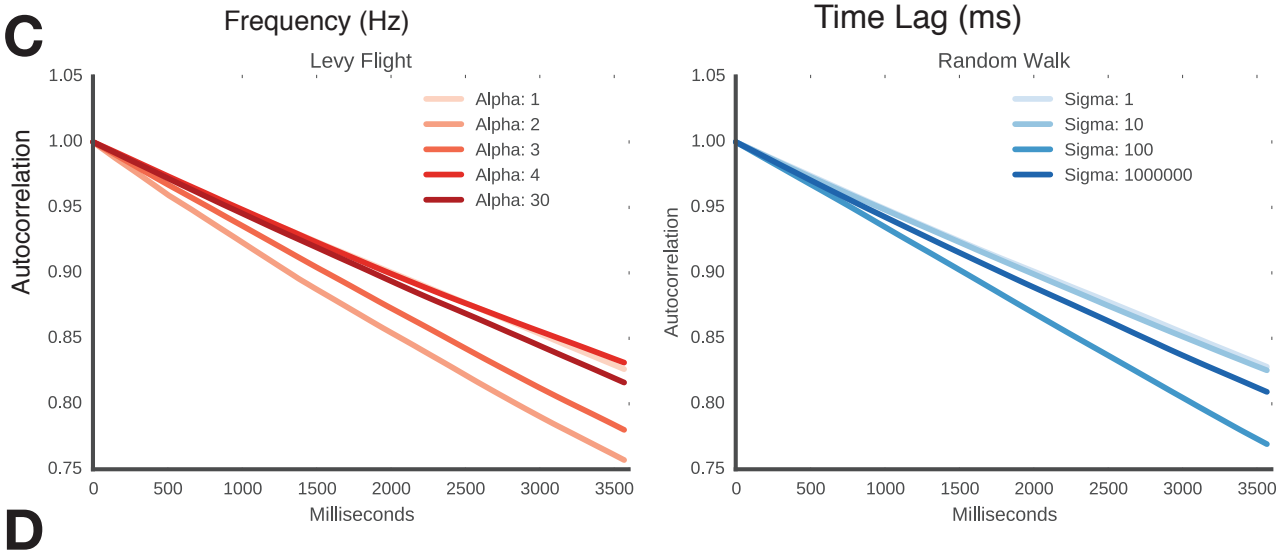
Wiener, N. (2013). Extrapolation, Interpolation, and Smoothing of Stationary Time Series, with Engineering Applications (Martino Fine Books).

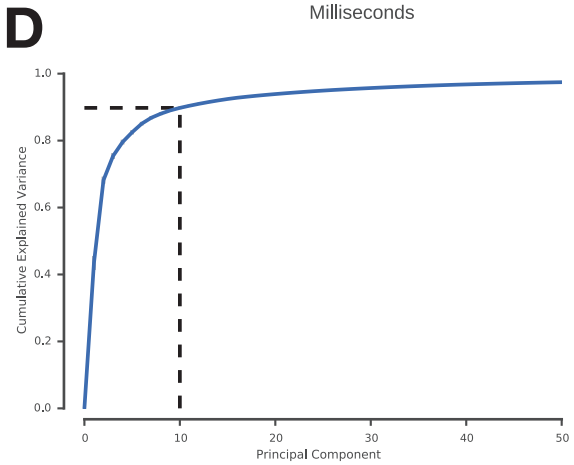

A
Record Raw Depth Frames on Microsoft Kinect
Select region-of-interest (ROI)
B
Calculate median raw background image
Crop Raw Images to ROI
C
Impute Missing Values
D
Calculate X,Y,Z position in millimeters for each pixel
X
Y
Z
E
Calculate median real-world coordinate background image
Resample pixel positions to an even grid in millimeters
F
Subtract background from real-world coordinate images
Mouse
ROI

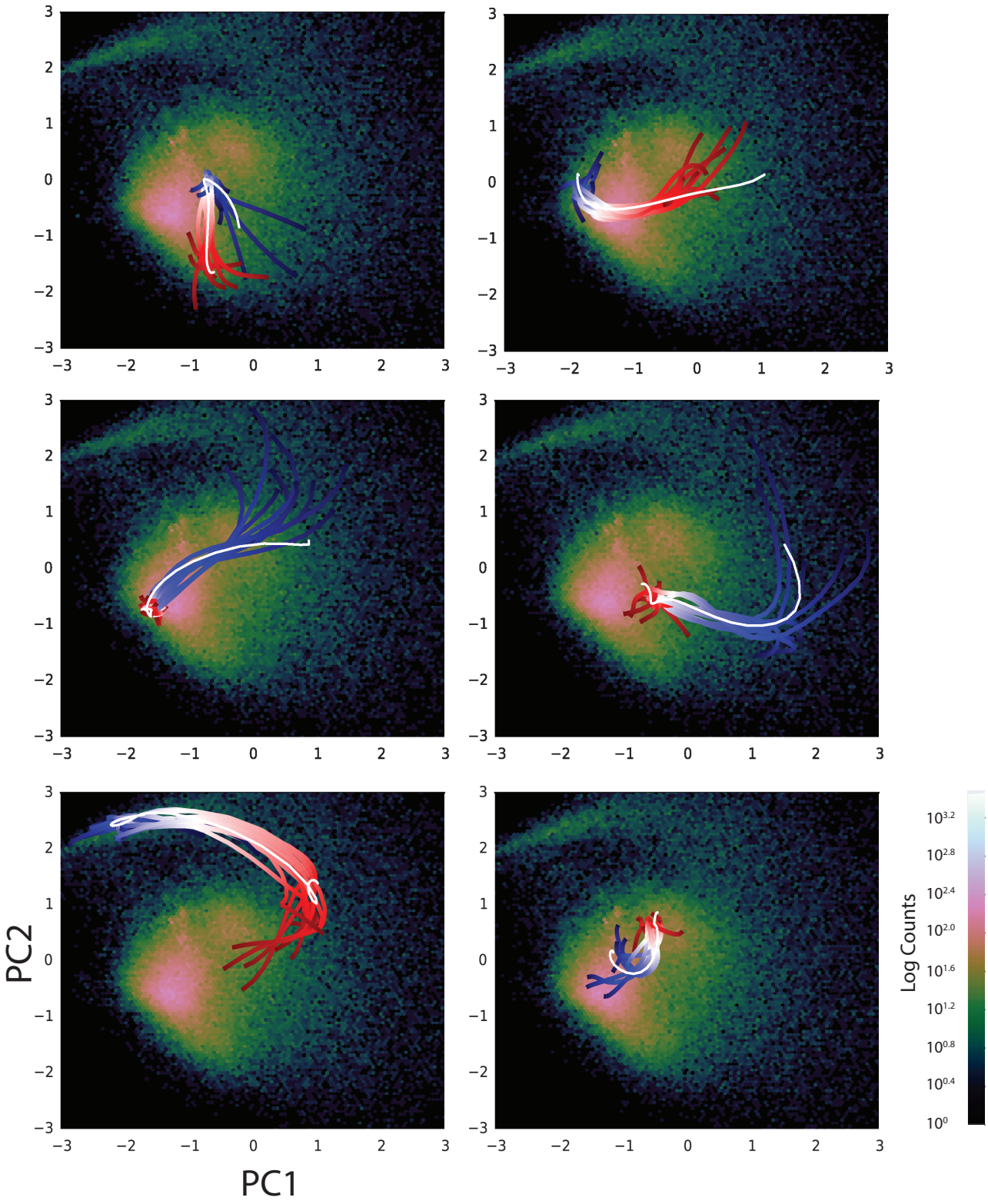

Background-subtracted Images
G
Per-frame artifact removal and iterative median filtering
H
Mouse Identification via blob detection
I
Image extraction
J
Center-of-mass correction
Scalar extraction
Orientation Correction
Ellipse Fitting
Artifact Removal
Mouse Identification
Pose Image, Unrotated
Pose Image, Orientated Corrected

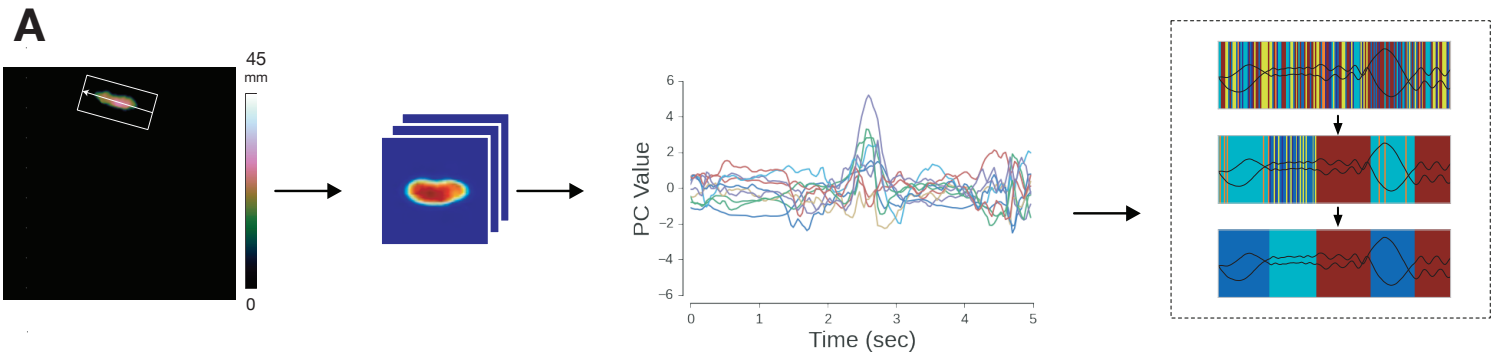

A
Live, Smoothed
Posed, Smoothed
Live, Unsmoothed
Posed, Unsmoothed
Dimensions
Random Projections
Spine Height
Raw Pixels
Velocity (cm/s)
+15
-15
45 mm
0 mm
5 sec


B
Spectrum Ratio
Frequency (Hz)
Autocorrelation
Time Lag (ms)
Random Proj.
OFA
TMT Odor
Blank Odor
RorB
Optogenetics
Posed


C
Levy Flight
Random Walk
Autocorrelation
Milliseconds
Alpha: 1
Alpha: 2
Alpha: 3
Alpha: 4
Alpha: 30
Sigma: 1
Sigma: 10
Sigma: 100
Sigma: 1000000


D
Cumulative Explained Variance
Principal Component


PC2
PC1
Log Counts
$10^{3.2}$
$10^{2.8}$
$10^{2.4}$
$10^{2.0}$
$10^{1.6}$
$10^{1.2}$
$10^{0.8}$
$10^{0.4}$
10^{0}



Preprocess Raw Imaging Data

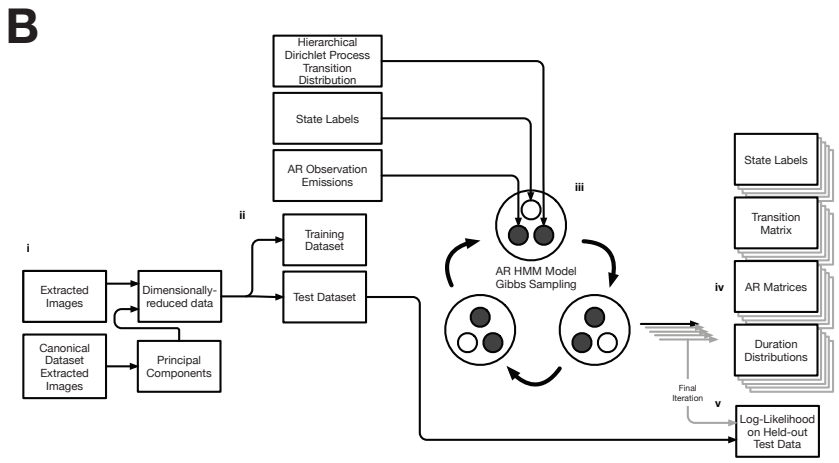
- Extract image of mouse
- Correct parallax artifacts
- Align mouse along axis of spine

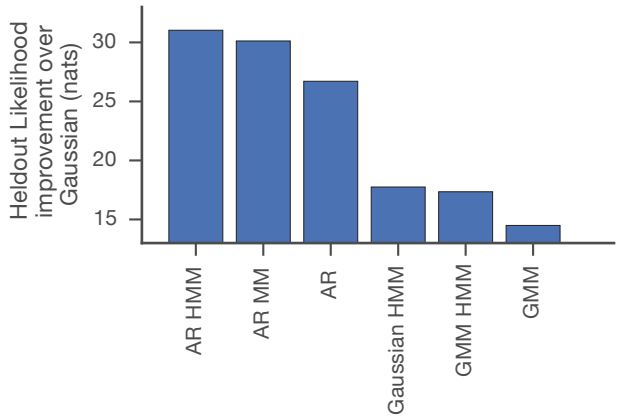
Dimensional Compression of 60x60 pixel data

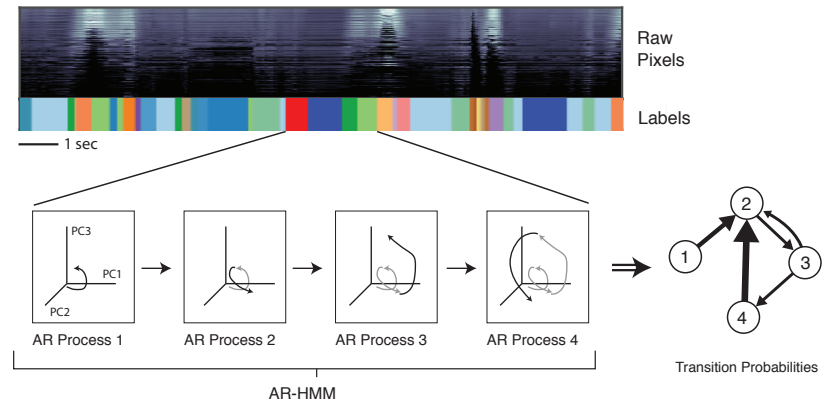
- Wavelet Transformation
- Principal components analysis
- Extract 10 PCs of pixel data

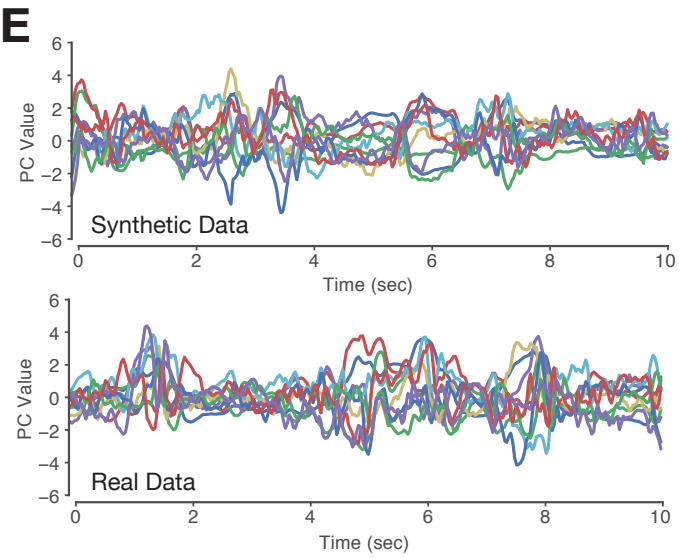
Submit PCA data to various models

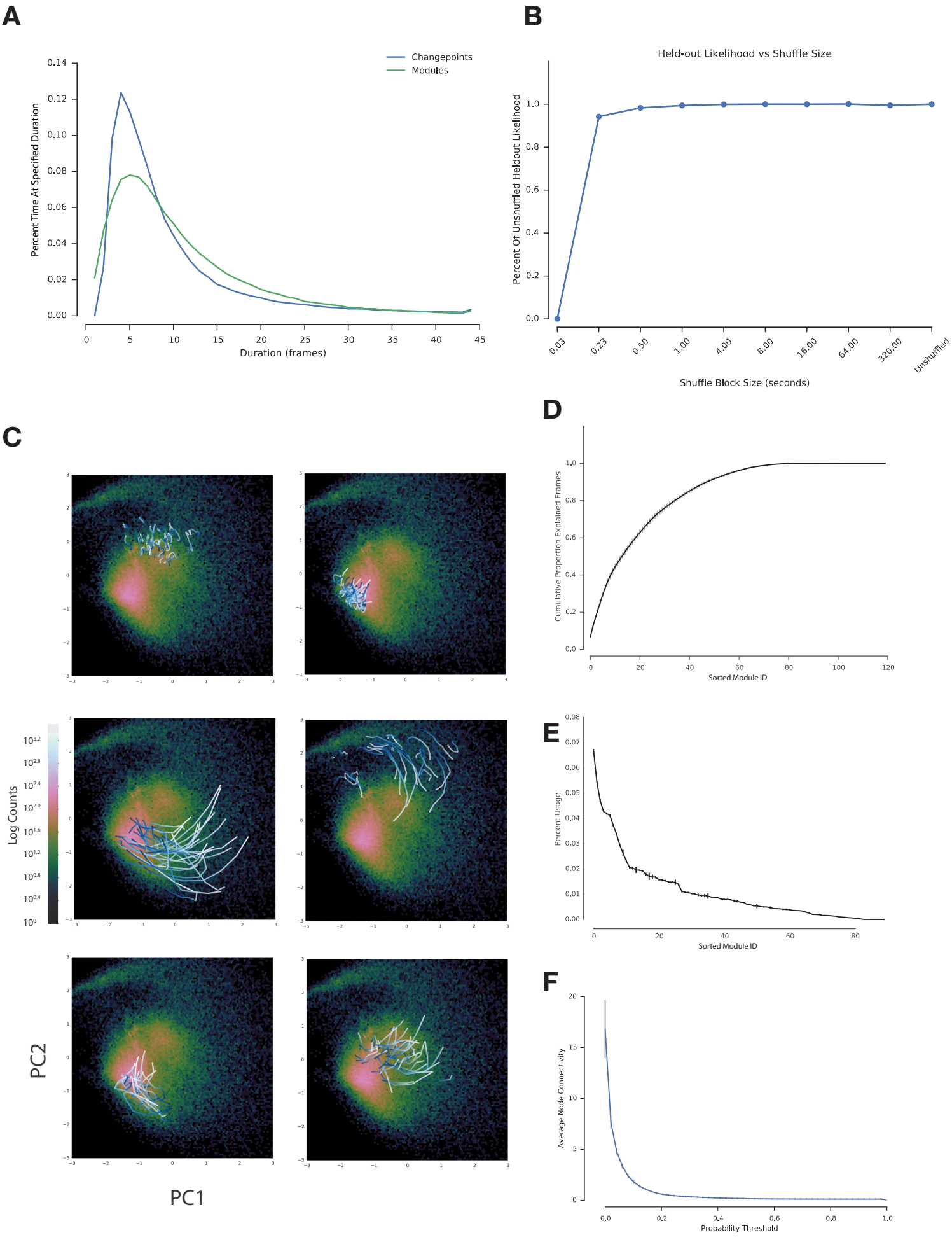
Model Fitting with Gibbs Sampling to Identify Behavioral Modules And Transition Statistics



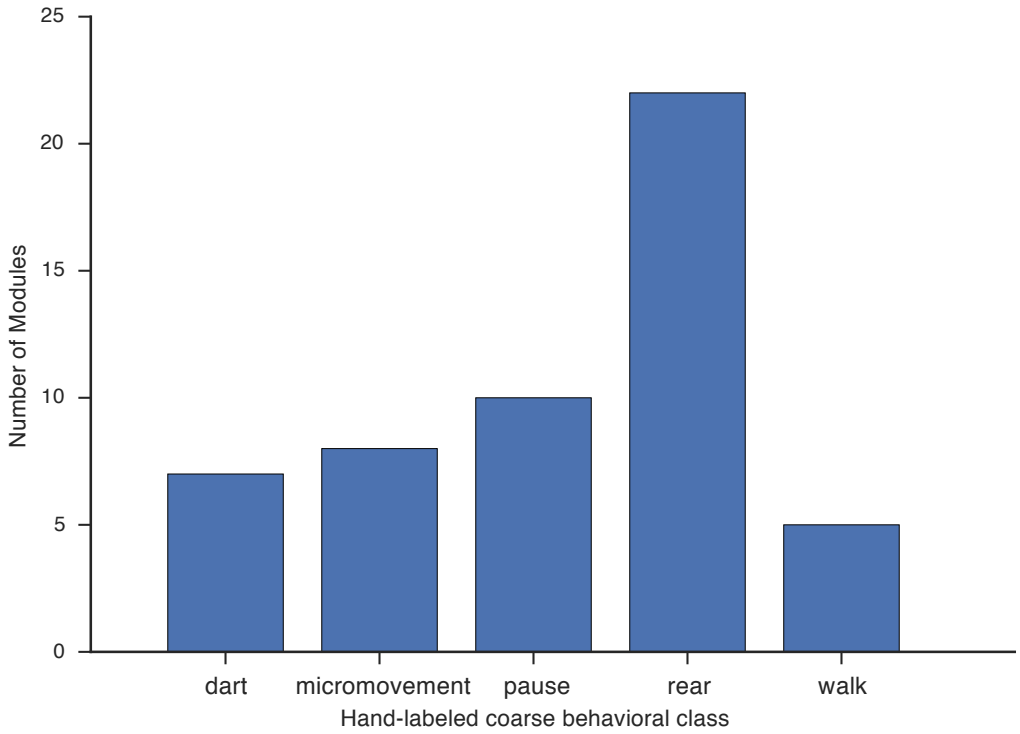




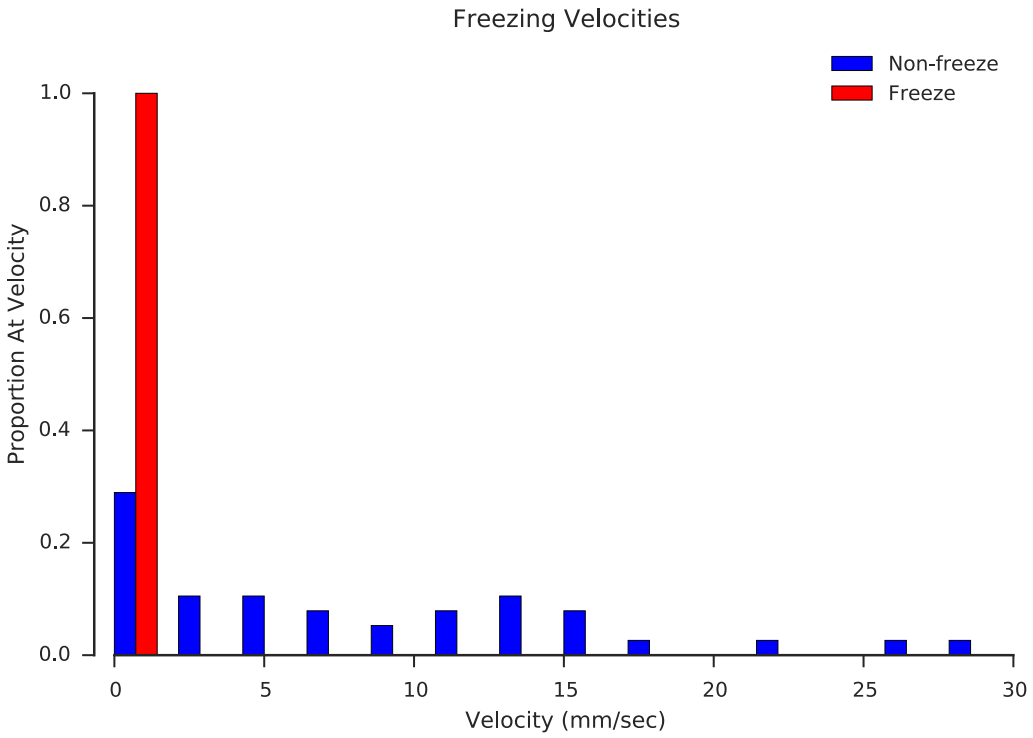



A
Changepoints
Modules
Percent Time At Specified Duration
Duration (frames)
B
Held-out Likelihood vs Shuffle Size
Percent Of Unshuffled Heldout Likelihood
Shuffle Block Size (seconds)
0.03
0.23
0.50
1.00
4.00
8.00
16.00
64.00
320.00
Unshuffled
C
Log Counts
PC2
PC1
D
Cumulative Proportion Explained Frames
Sorted Module ID
E
Percent Usage
Sorted Module ID
F
Average Node Connectivity
Probability Threshold

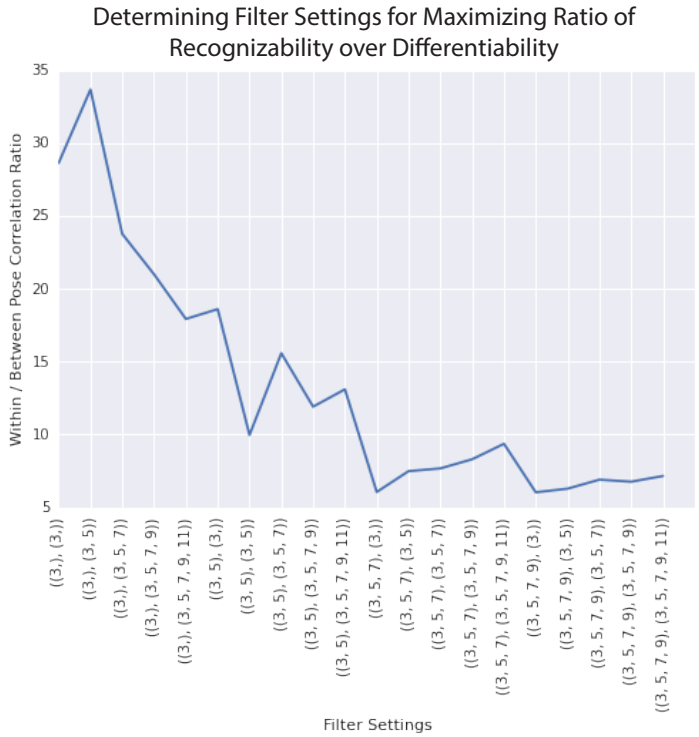
A


25
20
15
10
5
0
Number of Modules
dart
micromovement
pause
rear
walk
Hand-labeled coarse behavioral class

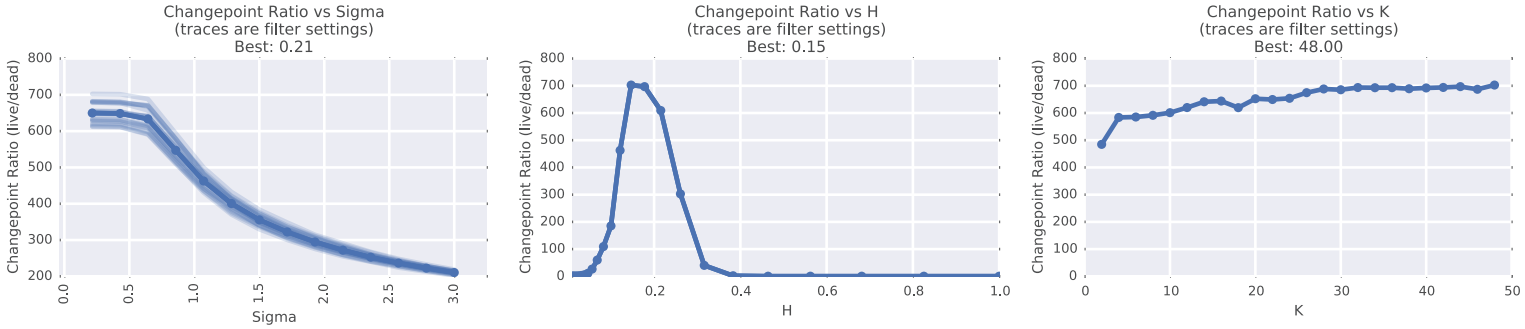
B


Freezing Velocities
Non-freeze
Freeze
1.0
0.8
0.6
0.4
0.2
0.0
Proportion At Velocity
0
5
10
15
20
25
30
Velocity (mm/sec)

A


Determining Filter Settings for Maximizing Ratio of Recognizability over Differentiability
Within / Between Pose Correlation Ratio
35
30
25
20
15
10
5
((3,), (3,))
((3,), (3, 5))
((3,), (3, 5, 7))
((3,), (3, 5, 7, 9))
((3,), (3, 5, 7, 9, 11))
((3, 5), (3,))
((3, 5), (3, 5))
((3, 5), (3, 5, 7))
((3, 5), (3, 5, 7, 9))
((3, 5), (3, 5, 7, 9, 11))
((3, 5, 7), (3,))
((3, 5, 7), (3, 5))
((3, 5, 7), (3, 5, 7))
((3, 5, 7), (3, 5, 7, 9))
((3, 5, 7), (3, 5, 7, 9, 11))
((3, 5, 7, 9), (3,))
((3, 5, 7, 9), (3, 5))
((3, 5, 7, 9), (3, 5, 7))
((3, 5, 7, 9), (3, 5, 7, 9))
((3, 5, 7, 9), (3, 5, 7, 9, 11))
Filter Settings

B


Changepoint Ratio vs Sigma
(traces are filter settings)
Best: 0.21
Changepoint Ratio (live/dead)
800
700
600
500
400
300
200
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Sigma
Changepoint Ratio vs H
(traces are filter settings)
Best: 0.15
Changepoint Ratio (live/dead)
800
700
600
500
400
300
200
100
0
0.2
0.4
0.6
0.8
1.0
H
Changepoint Ratio vs K
(traces are filter settings)
Best: 48.00
Changepoint Ratio (live/dead)
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
K