A FRAMEWORK TO IDENTIFY STRUCTURED BEHAVIORAL PATTERNS WITHIN RODENT SPATIAL TRAJECTORIES

FRANCESCO DONNARUMMA[1], ROBERTO PREVETE[2], DOMENICO MAISTO[3], ANDREA FUSCONE[2], MATTHIJS A. A. VAN DER MEER[4], CALEB KEMERE[5] & GIOVANNI PEZZULO[1]

ABSTRACT

Animal behavior is highly structured. Yet, structured behavioral patterns – or "statistical ethograms" – are not immediately apparent from the full spatiotemporal data that behavioral scientists usually collect. Here, we introduce a framework to characterize quantitatively rodent behavior during spatial (e.g., maze) navigation, in terms of movement building blocks or *motor primitives*. The hypothesis underlying this approach is that rodent behavior is characterized by a small number of motor primitives, which are combined over time to produce open-ended movements. We introduce a machine learning methodology – dictionary learning – which permits extracting motor primitives from rodent position and velocity data collected during spatial navigation and use them to both reconstruct past trajectories and predict novel ones. Three main results validate our approach. First, our method reconstructs rodent behavioral trajectories robustly from incomplete data, outperforming approaches based on standard dimensionality reduction methods, such as principal component analysis. Second, the motor primitives extracted during one experimental session generalize and afford the accurate reconstruction of rodent behavior across successive experimental sessions in the same or in modified mazes. Third, the number of motor primitives that our method associates to each maze correlates with independent measures of maze complexity, hence showing that the motor primitives formalism is sensitive to essential aspects of task structure. The framework introduced here can be used by behavioral scientists and neuroscientists as an aid for behavioral and neural data analysis. Indeed, the extracted motor primitives enable the quantitative characterization of the complexity and similarity between different mazes and behavioral patterns across multiple trials (i.e., habit formation). We exemplify some uses of the method to control for confounding effects (e.g., of maze complexity on behavior and reward collection), analyze habitual or stereotyped behavior, classify or predict behavioral choices as well as place and grid cell displacement in new mazes. Keywords: maze navigation; spatial trajectories; motor primitives; ethograms

* *Corresponding author (giovanni.pezzulo@istc.cnr.it)*
[1] *Institute of Cognitive Sciences and Technologies, National Research Council, Via S. Martino della Battaglia 44, 00185 Rome, Italy*
[2] *Department of Electric Engineering and Information Technologies (DIETI), Università degli Studi di Napoli Federico II, Naples, Italy*
[3] *Institute for High Performance Computing and Networking (ICAR), National Research Council (CNR), Via Pietro Castellino 111, 80131 Naples, Italy*
[4] *Department of Psychological and Brain Sciences, Dartmouth College, Hanover, NH, USA*
[5] *Electrical and Computer Engineering, Rice University, Houston, Texas*

1 INTRODUCTION

During cognitive tasks, such as spatial navigation, animals exhibit highly structured behavior. These behavioral patterns or "motifs" can indicate important latent dimensions, such as task complexity (e.g., movement variability) or the amount of training that the animal has received (e.g., movement stereotypy). Yet, a standard methodology to extract structured behavioral patterns from the set of data (e.g., position and velocity) that behavioral scientists usually collect during spatial navigation is still lacking.

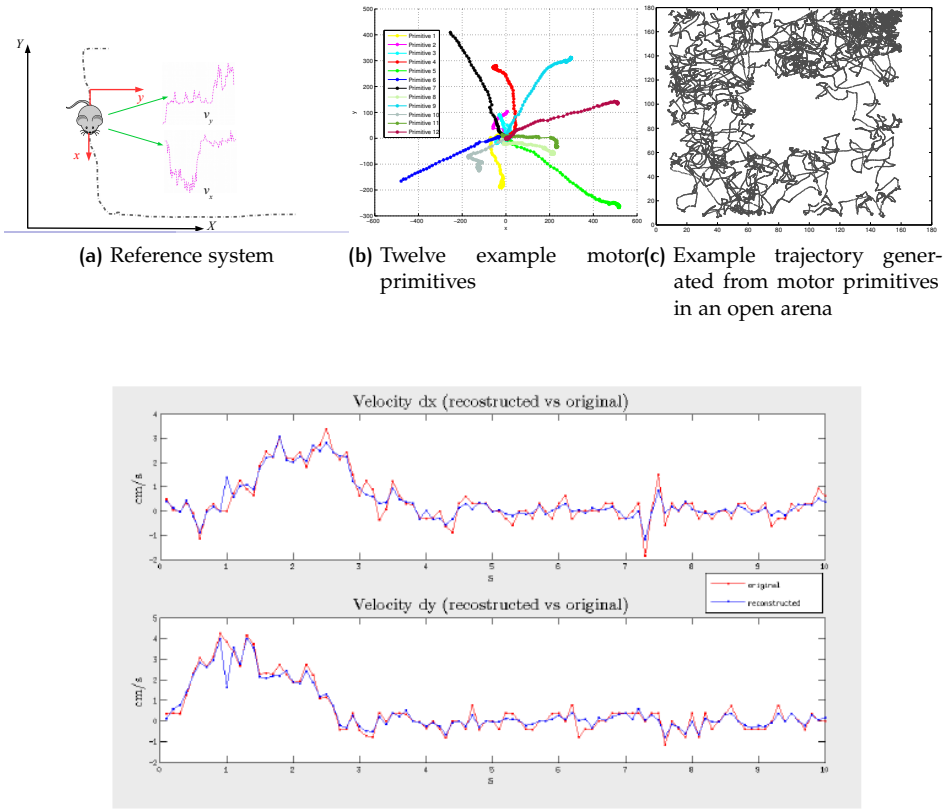
Here we present a computational framework that characterizes the latent structure – or a "statistical ethogram" – of rodent movement dynamics during spatial navigation, in terms of movement building blocks or motor primitives. The idea that motor control uses a combination of motor primitives is widely accepted in computational neuroscience [1, 2, 3, 4, 5, 6] but its application for the study of rodent spatial navigation is less common. An assumption of this approach is that even sophisticated (e.g., spatial) movements can be characterized in terms of a limited basis of dynamic motor primitives, which combine over time – such that, for each time frame, one or more motor primitives can be active. In other words, the brain may adapt motor primitives as fundamental units to organize complex (spatial or other) movements in time. Therefore, inferring the animal's motor primitives during movements can help understanding how the brain organizes complex behavioral patterns [7].

There is also a second, more practical reason to use the motor primitives approach to study animal movements (even if ultimately the brain uses a different organization). Characterizing animal behaviour in terms of its underlying motor primitives can be more practical than using the full spatiotemporal [x(t),y(t)] data and more indicative of underlying regularities (e.g., behavioural stereotypy), as demonstrated by the analysis of behavioral motifs in various animal, species such as Caenorhabditis elegans [8], fruit fly [9] and rat [10]. Although previous work has analyzed behaviors such as path stereotypy in terms of path similarity (e.g. [11]), such descriptive approaches cannot generate novel data. More recent unsupervised learning methods [12] or related approaches [13, 14, 15, 16] are more similar to our approach, but have not yet been applied to structured maze data.

In this study, we illustrate a novel computational approach to learn a dictionary (called V) of motor primitives from rodent trajectory data; and validate the approach by showing that it affords an accurate reconstruction of (novel) trajectories. Figure 1 shows a schematic illustration of the modeling methodology used in this paper. First, we extract spatial (x,y) and velocity (x,y) data from animal trajectories during spatial navigation, see Figure 1a. Second, we use these data to learn a *dictionary (V)* of motor primitives, using a *dictionary learning* [17, 18, 19] approach (see the Methods section). In this approach, primitives are spatiotemporal patterns that are recurrent in the animal's behavior and thus permit reconstructing it in a parsimonious way (e.g., one can reconstruct a large dataset of trajectories as a combination of a small number of motor primitives). Figure 1b shows a sample dictionary of 12 primitives extracted from data (note that most emerged primitives only last few seconds, see below). Here, only the spatial (not the velocity) components of the primitives are shown. These primitives can then be chained to simulate spatial trajectories, see Figure 1c for an example in an open arena. Furthermore, they can be used to "reconstruct" actual animal trajectories, see Figure 1d. In this latter case, by varying the parameters of the *dictionary learning* method – i.e., *sparsity* and *dictionary size* – it is possible to search for the best data fit / reconstruction; and therefore analyze which motor primitives (or their combination) better explain the trajectory data. The relevant parameters are *dictionary size* (i.e., how many primitives a dictionary can include) and two kinds of *sparsity*. The former, *dictionary sparsity*, implies that the same motor primitive should only be active for a limited portion of the whole trajectory – in this study, lasting a maximum of 10 seconds. The second, *co-*

efficient sparsity, implies that at each time step, movement can be reconstructed by superimposing at most few primitives – and possibly just one. Please refer to the Methods section for additional details.

The possibility to model, predict and reconstruct spatial trajectories in terms of their underlying motor primitives opens the doors to a number of potential applications of the framework, including the possibility to discover sequential structure, repeating patterns and stereotypy in behavior, and predict animal choices – which we discuss and exemplify in the final section.



(a) Reference system

(b) Twelve example motor primitives

(c) Example trajectory generated from motor primitives in an open arena

(d) Velocity profiles of real data vs. reconstructed

Figure 1: Schematic illustration of the approach. Panel 1a: the rodent-centered coordinate system used in as input for the dictionary learning model. The dotted black line is the actual animal trajectory. For each time, we extracted from data a 2-D (x, y) spatial position. Note that our reference system is not anchored to the maze. Rather, at each time point, x represents the last direction of the animal. The resulting, animal-centered reference system permits deriving different primitives for different orientations of the animal. Furthermore, we extracted animal velocity in x and y dimensions. Panel 1b: Twelve sample motor primitives extracted from navigation data. Panel 1c: the same motor primitives used to generate (simulate) trajectories in an open arena. Panel 1d: example reconstruction of real rodent data using motor primitives (here, only the velocity in x and y axes is reconstructed). Blue is the actual velocity profile over time, red is the reconstructed profile. See the main text for explanation.

2 RESULTS

2.1 Dataset

We conducted a series of experiments to validate our methodology, using a dataset of rodent movements collected in Van der Meer Lab (Dartmouth College). The dataset includes data from 3 rats (R01, R02, R03). Each rat experienced a different maze every day, for a total of 8 mazes / days. Maze navigation consisted of shuttling repeatedly between two goal (reward) locations, placed at the ends of a U-shaped track. For each maze, data were collected in three different phases. In the first and second phases, the animal was able to navigate only the U-shaped part of the track. In the third phase, the animal had access to two new paths in the same maze – one of which was a shortcut between the two reward locations, and the other a dead end. Figure 2 shows the eight mazes and the trajectories of animal R01 in the three phases: green (phase 1), yellow (phase 2) and grey (phase 3). Table 1 shows the overall duration of the navigation episodes of animal R01 (across multiple trials, see below) for each phase and day.

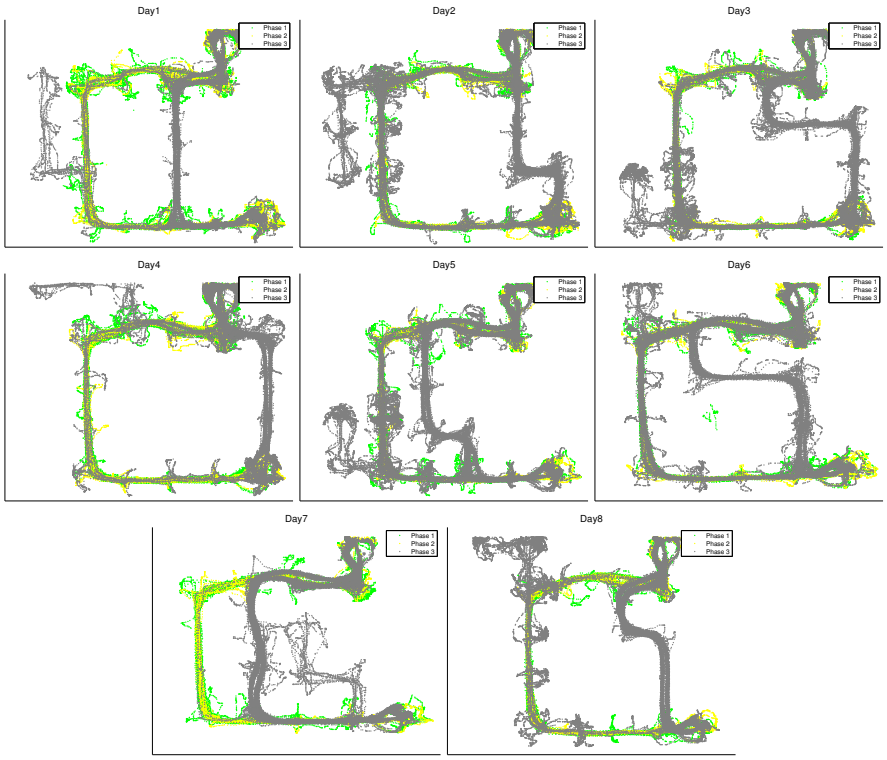


Figure 2: Dataset: trajectories of animal R01 across 8 days, 3 phases every day. See the main text for explanation.

R01 (min)	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8
Phase1	10.08	10.05	8.02	8.13	9.88	8.48	8.41	8.10
Phase2	20.33	20.37	15.02	20.55	20.33	21.12	20.24	20.04
Phase3	25.18	46.10	45.05	42.68	52.21	45.82	40.43	55.04

Table 1: Duration in minutes of the different phases, rat R01

2.2 Experimental results, animal R01

In all the analyses below, we focus on trajectory data of animal R01 for illustrative purposes; see the Supplementary Material for data on animals R02 and R03, which show the same pattern of results as animal R01.

2.2.1 *Building the dictionary of Primitives using SRSSD*

The methodology to build and select the dictionary of primitives to be used in our subsequent analyses includes three main steps.

The first (preparation) step of our methodology consists in dividing the database of rat trajectory data into three sets: a *training set*, which includes data from phases 1, 2 and 3 of day 1; a *validation set* and a *test set*, both including (non-overlapping) data from phase 3 of days 1 to 8.

To obtain the actual training, validation and test sets, we split the aforementioned rat trajectories into "patches" (i.e., portions of animal trajectories in x,y), lasting a maximum of 5 seconds each. Then, we randomly selected 10000, non-overlapping patches to compose each of the three sets. Patches with missing data (e.g., cases in which the animal goes out of the video camera frame) were excluded.

The complexity of these trajectory data can be appreciated by considering that a principal component analysis (PCA) shows that 58 principal components are necessary to explain about 90% of their variance, see Figure 3.

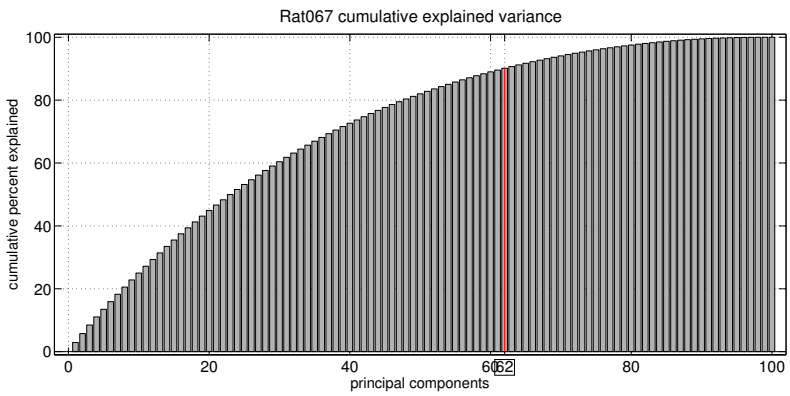


Figure 3: Explained Variance: In the dataset, 58 principal components are necessary to explain about 90% of the variance of the data.

SRSSD procedure depends on two parameter of sparsity (dictionary sparsity and coefficient sparsity) and outputs the best couple V (dictionary of motor primitives) and U (coefficients) that reconstruct the data (see Algorithm 1). Thus, the second (learning) step of the methodology consists in deriving candidate SRSSD dictionaries of motor primitives, which reconstruct accurately the *training set* data. In this step, we select the best 600 (6x10x10) candidate dictionaries) i.e., the best 600 dictionaries obtained in reconstructing training data by varying 6 different dictionary sizes (with 25, 50, 75, 100, 125, and 150 motor primitives, respectively), 10 levels of dictionary sparsity and 10 levels of coefficient sparsity,

To measure how well the 600 candidate SRSSD dictionaries reconstructed the *training set* data, we adopted a methodology that is analogous to the *missing pixel method*: we canceled out some parts (from 10% to 90%) of the trajectories that compose the training set and evaluated how well the learned dictionaries permit to reconstruct these missing parts (see the Methods section for an explanation of how the reconstruction error RMS is calculated). Figure 4(left) shows that the solutions found by SRSSD (for 100 dictionaries of size 150, i.e., the size that gave the best results) during this learning phase cover extensively the space of solutions, suggesting a good choice of parameters.

Consequently, in the third (validation) step of the methodology we select the best 30 SRSSD dictionaries V amongst the aforementioned 600 candidates. To do this we use sparse coding reconstruction (see Algortihm 2) fixing V and obtaining the couple V and U that better reconstruct the *validation set* data now varying 5 levels of "noise" and 10 levels of coefficient sparsity. For this, we calculated the reconstruction error of the dictionaries constructed during the previous (learning) step and of PCA, on the validation set. While we conducted the validation on all the 600 dictionaries considered above, we show the results of the 100 dictionaries of size 150 (i.e., the size that gave the best results during the learning step). Figure 4(right) shows the reconstruction results in the validation phase of these 100 dictionaries, with 10 levels of coefficient sparsity and 5 levels of "noise" (0.1, 0.3, 0.5, 0.7 and 0.9) – corresponding to the percentage of "missing" trajectory data (10%, 30%, 50%, 70% and 90%) to be reconstructed.

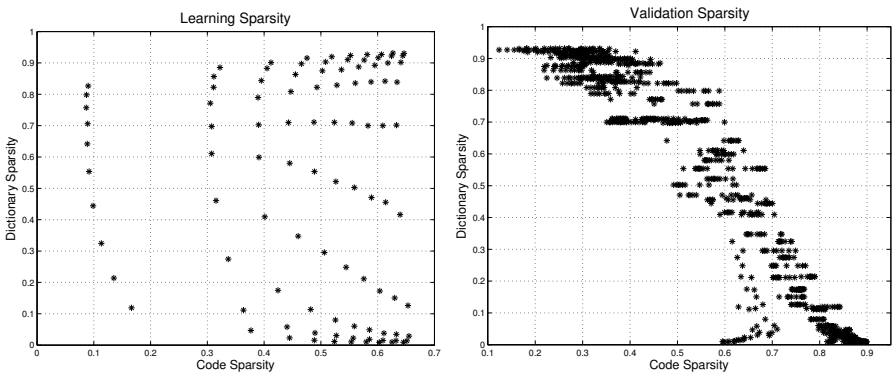


Figure 4: SRSSD method applied to navigation data from animal R01. Left: learning phase. Right: validation phase. Results are shown for the best dictionary (i.e., having lower RMS in the test set / validation phase) of 150 motor primitives, with 10 values of coefficient sparsity (found during the learning phase) and 5 levels of noise (0.1, 0.3, 0.5, 0.7, 0.9).

The results of the validation step are shown in Figure 5. In the figures, PCA results are shown with a horizontal red line, whereas results using various dictionaries are shown in blue. The PCA results were computed by comparing the reconstruction error with various principal components (25, 50, 75, and 100) and selecting the best (i.e., with 100 principal components). The comparison illustrates that for each level of noise, various dictionaries have lower reconstruction error (RMS) than PCA. We found that higher noise levels decrease reconstruction accuracy (which is expected), but also that SRSSD is more robust and outperforms PCA across all levels of noise, see Figure 6.

We use the best 30 SRSSD dictionaries for our subsequent analyses.

2.2.2 *Reconstruction across consecutive days using the best 30 SRSSD dictionaries*

We tested whether the best 30 SRSSD dictionaries V identified using the above procedure afford an accurate reconstruction of novel data, from the *test set*. More specifically, we were interested in assessing whether a learning procedure that only considered data from day 1 (i.e., *training set*) permitted reconstructing non-overlapping data from the third phases of days 1-8 in our dataset (i.e., *test set*). Note that the reason why the *test set* only includes the third phases of each day is that this is when, in the rodent experiment, novel portions of the maze appear. Data on the third phases of each day are this ideal to test whether our approach generalises to novel and unforeseen situations. Furthermore, we asked whether the motor primitives required to reconstruct animal trajectories in a given day capture essential characteristics of the day's task, such as maze complexity or stereotypy of behaviour.

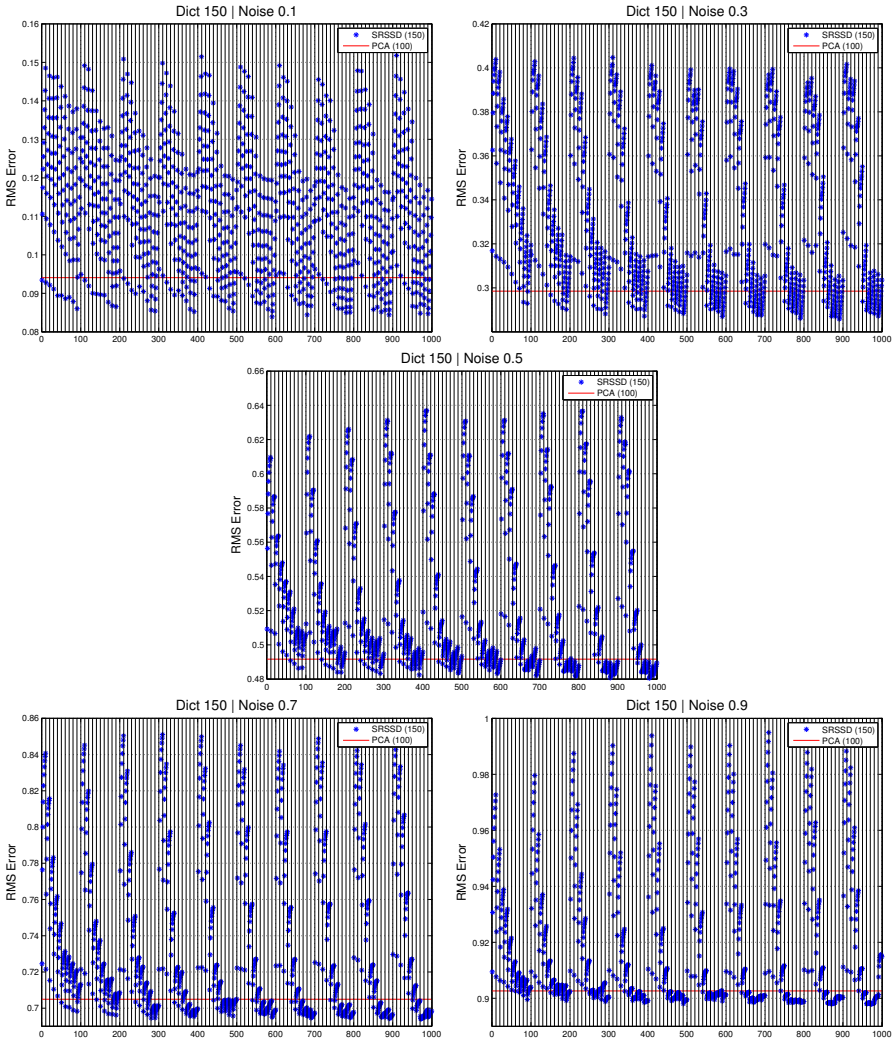


Figure 5: Comparison between the reconstruction (RMS) error of SRSSD (blue dots) and PCA (red line) across various levels of noise. Each figure shows 100 dictionaries found during the learning step. We remind that each dictionary has 2 kinds of sparsity: dictionary sparsity (which is fixed during the learning phase) and coefficient sparsity (which is varied here); see the Methods section. Here, each of the 100 dictionaries (with its fixed atom sparsity) is tested with 10 different values of coefficient sparsity. Hence, for each figure, each of the 1000 asterisks corresponds to a unique combination of dictionary and coefficient sparsity. The reconstruction error for all the asterisks that appear under the red line is better than PCA.

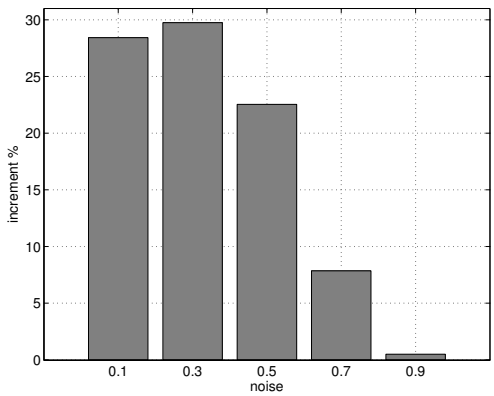


Figure 6: Increment of performance using the SRSSD approach over the PCA approach, with five different levels of noise.

Figure 7 shows the reconstruction results for the best 30 SRSSD dictionaries. There are several elements to be appreciated. First, reconstruction error is lower in days 1, 2 and 8. The low reconstruction error in day 8 indicates that the method generalises well across distal days. The visual inspection of the eight mazes suggests the importance of spatial similarity between mazes (e.g., mazes 1 and 8 are rather similar). Furthermore, it emerges from the results that days 4 and 8 are those with higher coefficient sparsity, implying that less motor primitives are required for a correct reconstruction.

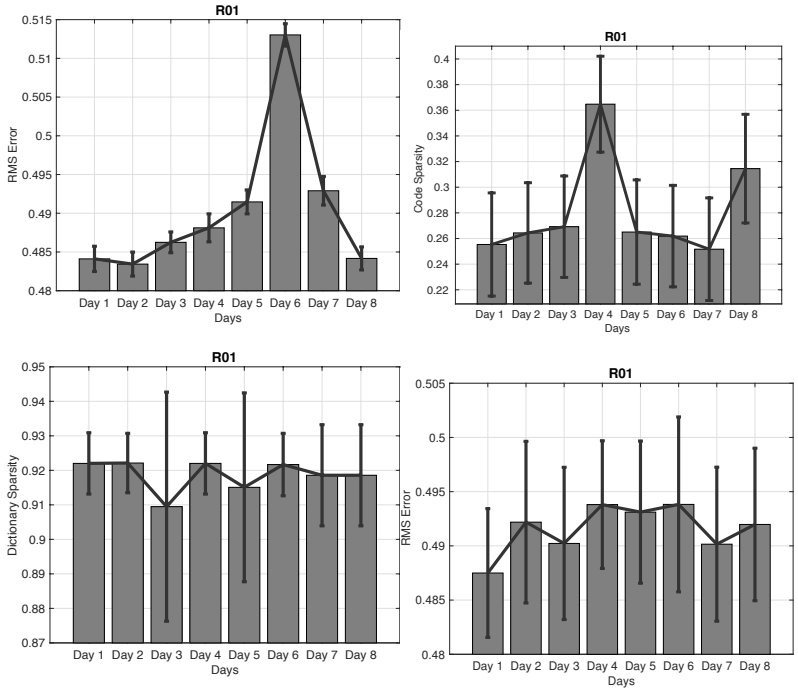
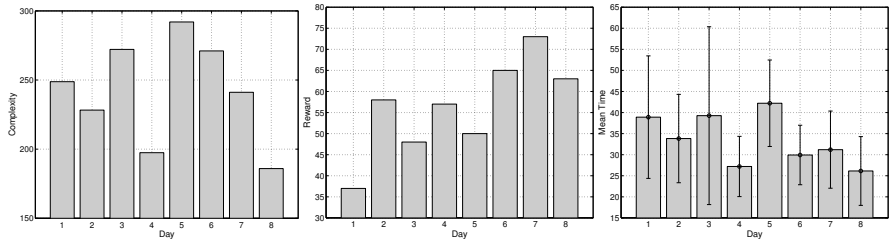


Figure 7: Reconstruction across 8 days (after learning only from data of day 1. Noise is 0.5 and the best 30 SRSSD dictionaries found on day 1 are used. (a): RMS Error. (b): coefficient sparsity. (c) dictionary sparsity. (d) Baseline performance: reconstruction of Phase 1.

To understand why this is the case, we conducted a further analysis of task demands and complexity, using various measures, see Figure 8. The first measure we considered is spatial complexity of the maze, which we assessed by dividing the maze in bins, counting how many bins the animal occupies for each trial of phase 3, and then averaging (i.e., dividing this number by the number of trials); see the Supplementary Material for Figures illustrating the results of this process. High spatial complexity (or entropy) implies that the animal has explored and occupied all the available positions, while low spatial complexity means that the animal followed stereotyped trajectories. Note that there are multiple ways to construct spatial complexity [20, 21]. In our approach, complexity depends on behavior (similar to [22]) – or better, on a graph constructed on the basis of the actual trajectories of the animal, rather than a fixed graph that only includes spatial dependencies.

As shown in Figure 8a, maze complexity is lower in days 4 and 8. We next considered how many times the animal collected a reward (Figure 8b), but this measure does not correlate with measures extracted from SRSSD. Finally, we considered the mean time the animal needed to complete a trial, see Figure 8c. Required time is lower in days 8 and 4 – which was expected, given that the previous spatial complexity analysis revealed more stereotyped behavior in the same days. Overall, this analysis reveals that maze complexity in days 4 and 8 is lower, and the animal behavior is faster. It thus supports the idea that the motor primitives formalism we adopted captures essential task components such as its complexity.



(a) Spatial complexity of the maze

(b) Rewards obtained

(c) Mean time to complete the trial

Figure 8: Measures of task demands and complexity. (a): Spatial complexity of the maze. (b) Rewards collected. (c) Average time required to complete a trial and secure a reward.

2.3 Example applications of the method in the generation, description and classification of behavior

Animal behavior is highly structured. Behavioral scientists and neuroscientists often deal with complex datasets of behavioral data – e.g., the full spatiotemporal x(t), y(t) data – that are not convenient for asking questions like: how stereotyped is the animal's behavior? How well can future behavior be predicted by past behavior? How does behavior evolve with experience on the task, within and across sessions?

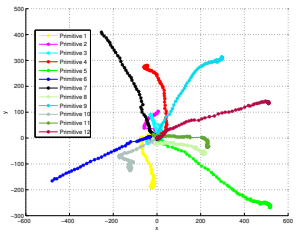
One appeal of the motor primitive code is that it provides a description of rat behavior that is more compact than the full spatiotemporal [x(t),y(t)] data yet richer than a simple mono-dimensional descriptor [e.g., speed(t)] – and lends itself to a number of applications. Our method based on motor primitives permits estimating the building blocks of a "computational ethogram" that captures the main behavioral regularities (i.e., repeated spatiotemporal sequences that are highly predictable within-sequence) and the temporal dependencies between them (e.g., transitions or contingencies between sequences). This method can be used in a number of ways.

2.3.1 *Generating simulated behavior on novel mazes and associated neuronal responses*

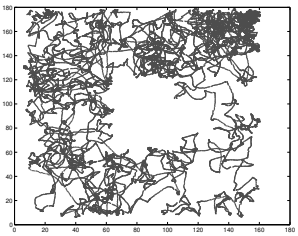
One possible application would be generating simulated behavior on various novel mazes. Simulated trajectories can be helpful in estimating the to-be-expected complexity of behavioral data. Furthermore, they can be fed as input to other components to predict hippocampal place cell activity, thus affording power analyses. One example is shown in Figure 9. We used the computational models of [23, 24] to simulate the firing of example grid and place cells during uniform exploration (Panels 9c and 9e, respectively) versus partial exploration of the environment following a simulated trajectory (Panels 9d and 9f, respectively). This example illustrates the possibility to use simulated trajectories to predict neuronal firings in novel environments. Note that in the above examples, the trajectories are generated by chaining motor primitives randomly, mimicking uniform exploration. Future studies may consider extensions of this method that learn (flat, hierarchical or context-dependent) transition probabilities between primitives and can generate more contextually-adequate trajectories.

2.3.2 *Assessing behavioral stereotypy after learning.*

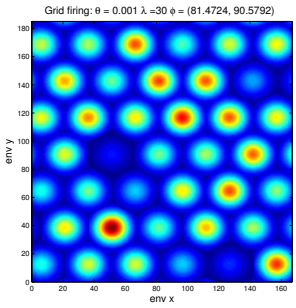
Another application is testing stereotypy of behavior (as a possible index of habitization), by looking at coefficient sparsity during learning. Stereotypy of behavior can be tested by considering whether coefficient sparsity increases during learning, indexing the fact that the animal is using fewer primitives (and hence a more restricted and stereotyped behavioral repertoire). As an example of this analysis, we



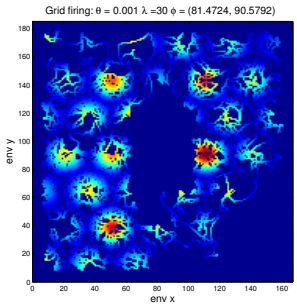
(a) Example primitives



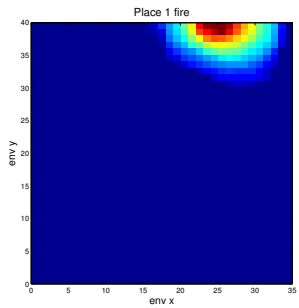
(b) Sample Trajectory



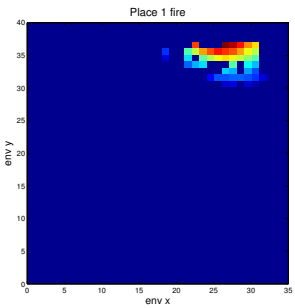
(c) Sample uniform exploration Grid response



(d) Sample path exploration Grid response



(e) Sample uniform exploration Place response



(f) Sample path exploration Place response

Figure 9: Graphical illustration of potential applications of the analysis. Motor primitives extracted from one maze (e.g., Panel 9a can be used to simulate animal behavior in novel mazes, e.g. the open arena shown in Panel 9b. Furthermore, one can simulate grid cells (Panel 9d) and place cells (Panel 9f) that would be expected if the animal follows the simulated trajectories of Panel 9b, as opposed to those that would be expected if the animal explores the open arena uniformly (Panels 9c and 9e). See the main text for explanation.

considered how coefficient sparsity changes while animal R01 learns to navigate the basic U-shaped maze (i.e., during phase 1 of day 1). For this, we have split data of phase 1 of day 1 in 5 intervals of the same length, removed the two extremes (1 and 5) and the middle interval (3), and compared intervals 2 vs. 4, corresponding to initial and final phases of learning, see Figure 10. Our results show that coefficient sparsity of R01 increases as the animal learns the task – indicating that the animals are indeed using less primitives and a more stereotyped behavior. Assessing whether this method is potentially more robust than measures of path stereotypy [25, 11] or simple descriptions of x(t), y(t) data like Fourier decomposition [26] is beyond the scope of this work. Yet, in general, describing spatiotemporal data in terms of (movement) primitives is considered to be more noise-tolerant in motor control [3, 4, 5, 6].

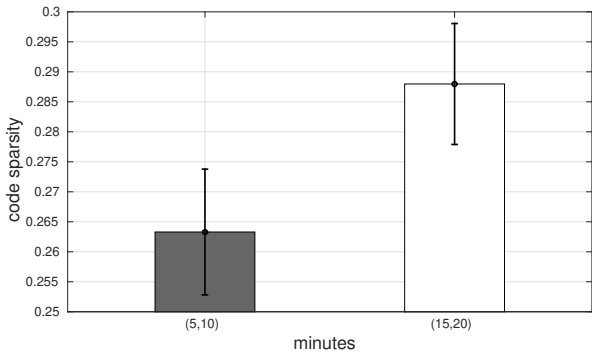


Figure 10: Coefficient sparsity of R01 increases in phase 1 of day 1, while the animal learns navigating the U maze, thus revealing stereotypy of behaviour. We report the comparison between the 2nd and 4th intervals of the same length, see the main text for explanation.

2.3.3 *Classifying and predicting behavior.*

Our approach based on the identification of motor primitives can also be used to classify and predict animal behavior and choices. To exemplify this, we performed an experiment aimed to study whether motor primitives extracted during different trials of phases 3 of each day permit to classify whether in these trials animal R01 follows the U-path or the shortcut. For each day, we randomly select 1000 trajectories (500 Shortcut and 500 U-path trajectories) and apply the reconstruction method, for each of the 30 best dictionaries, in order to find sparse coefficients. We then use the coefficients to classify U-path versus shortcut trajectories (which we had previously labelled, see examples in the Supplementary Materials). For this, we perform a classification test on the set of 1000 trajectories, using a linear support vector machine (SVM) classifier in a k-fold cross validation (with k=5). We compute mean and standard deviation over the dictionaries the accuracies. Figure 11 shows that this method affords very accurate classification across all the 8 days. Note that this result does not trivially depend on the fact that the animal occupies different x-y coordinates during U-path versus shortcut trajectories. The motor primitives are agnostic about maze-centered x-y coordinates, as their reference system is centered on the animal, see Figure 1a.

The last bar of Figure 11 (labelled Day1-8) shows the results of a second experiment, in which we used a single dataset of 8000 samples of trajectories (1000 for each day, resulting in 4000 U-path and 4000 shortcut trajectories, combined together). Note that while the U-path remains constant across days, the shortcuts change every day, making this experiment potentially more challenging than the former. Again, we performed a k-fold cross validation test using a linear svm classifier and we computed accuracies over the 30 dictionaries coefficients. Even in this second experiment, our method affords very accurate classification of U-path

versus shortcut trajectories. These results illustrate that motor primitives extracted through our method capture behaviorally relevant regularities that are informative (for example) about animal trajectories and choices.

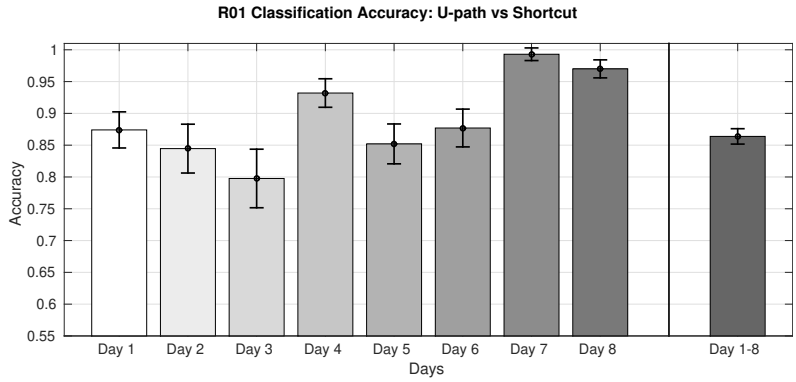


Figure 11: Classification of trajectories (U-path or shortcut) of phase 3, day 1, animal R01

3 DISCUSSION

We introduced a domain-general method to extract latent structure – a "statistical ethogram" – from rodent spatial navigation data (i.e., position and velocity), in terms of movement building blocks or motor primitives. Our experiments on rodent trajectories in eight mazes show that the dictionary learning (SRSSD) method outperforms principal component analysis (PCA) across all the noise levels we tested, with appreciable results over 0.5 noise, thus speaking in favor of the robustness of the methods. Furthermore, our results show good generalization across eight consecutive days. Despite we only used data from day 1 in the training set, the reconstruction of trajectories was not degraded in the successive seven days, suggesting that the motor primitives approach extracts regularities that are persistent over time. These regularities correspond to spatiotemporal patterns of movement, which can include for example rapid forward movements or rotations, which are repeated multiple times during the maze navigation and may be thus revelatory of an underlying modular organisation of behaviour.

Our further analysis indicates that our method based on motor primitives characterizes well task regularities such as spatial complexity and movement variance in terms of (coefficient) sparsity constraints. This makes intuitive sense, as navigation scenarios in which the animals' movements had lesser variance (e.g., or days 4 and 8 in animal R01) require less motor primitives (i.e., allow for more sparsity). It is also reassuring to notice that, although our method allowed the construction of motor primitives of up to 10 second, the primitives that actually populate the best dictionaries are of about 1 second, which is a more realistic time constraint for biological movement.

This is, to the best of our knowledge, the first framework that applies the concept of motor primitives – which is popular in motor control and computational neuroscience – to a data-driven analysis of rodent spatial navigation data. The method introduced here is based on 2D position data, which could be seen as limited compared to more sophisticated (3D camera, accelerometer) approaches [12]. However, 2D data is more widely available in the vast majority of experiments.

The method we have devised can be used by behavioral scientists and neuroscientists in behavioral and neural analyses. Some example uses that we have shortly illustrated in this paper are controlling for confounding effects (e.g., of maze complexity on behavior and reward collection), analyzing habitual or stereotyped behavior, classifying and predicting animal choices, and predicting place and grid cell

displacement in novel mazes. The results we reported illustrate the feasibility of the method, which can be proficuously added to the toolbox of behavioral scientists and neuroscientists to improve their ability to characterize quantitatively and understand animal behavior.

Besides, the method can be used for other purposes, such as for example to support the analysis of neuronal data or to design experiments that test the neuronal underpinning of motor primitives for spatial navigation. In decoding analyses, tuning curves (aka, the encoding model [27]) are typically estimated from those epochs during which the animal is running, which in turn is estimated using a running speed cutoff. However, the cutoff cannot clearly distinguish between different situations, e.g., grooming, directed running towards a goal or performing exploratory moves. Distinguishing these different behaviors is important as they modulate neural firings, above and beyond spatial position. The method presented here can distinguish different movement primitives (or even behavioral patterns or modes) and has thus the potential to improve our ability to relate neural activity to behavior, as done e.g. in decoding analyses. Furthermore, the motor primitives extracted from behavior may be characterized neurally. Studying the potential neuronal correlates of (spatial) motor primitives remains an important objective for future research.

Finally, the method can be extended to reveal and study more complex patterns of behavior than those considered here. Note that the method we have described does not try to estimate sequential dependencies between primitives. However, it would be trivial to compute those dependencies (e.g., a probability distribution $P(pi|pj)$ that a primitive pi follows a primitive pj) from training data, in order to estimate the most likely (or most surprising) transitions. It would be then possible to derive measures of path novelty, which would be related to (relatively) unpredictable transitions; or to study sequential (possibly, planning-related) patterns of behavior. These and other potential applications of the method remain to be tested in future studies.

4 MATERIALS AND METHODS

To characterize and extract motor primitives from trajectory data (i.e., position and velocity) during spatial navigation we adopt a *dictionary learning* approach [17, 18, 19]. Dictionary learning consists of supervised learning of a set of primitives (or atoms) that can be linearly superimposed to represent the elements of the dataset (e.g., rodent spatial trajectories). Starting from this initial idea, our algorithm is an extension particularly suitable for this kind of data because it incorporates two assumptions that are particularly useful when modeling spatial trajectories.

Firstly, it enforces *sparsity* at the level both of atoms and of coefficients: the former means that the same motor primitive is active just for a limited portion of the whole trajectory (lasting a maximum of 10 seconds, in this study), while the latter implies that at each time step, most few superimposing primitives – and possibly just one – can reconstruct movement. Secondly, dictionary learning assumes that motor primitives are contiguous; in this way, each motor primitive corresponds to a small part of a trajectory, and multiple motor primitives can be chained to form a complete trajectory. These two assumptions are lacking in other widespread methods, such as Principal Component Analysis (PCA) [28].

Another advantage of our approach over PCA is that the length of the primitives does not need to be predefined or fixed, but it is automatically determined by the algorithm, allowing for more flexibility depending on the particular application case.

4.1 Dictionary learning and the SRSSD approach

Recent machine learning approaches [29, 30, 31, 32, 33, 34, 35] represent signals as linear combinations of a large number of elements (*primitives* or *atoms*) — usually collected in sets called *dictionaries* — whose coefficients are computed using prior information encoded in penalization terms defining a particular kind of a minimization problem. These approaches are informatively grouped into two classes:

(i) ***Sparse atoms.*** This is the case when each atom involves just a small number of the original variables (see, for example, Sparse-PCA [29], sPCA-rSVD [36] and Structured-Sparse-PCA [37]);

(ii) ***Sparse coding.*** In this case, an *overcomplete* set of atoms is learned from the data, but the approximation of each signal involves only a restricted number of atoms. Hence, signals are represented by sparse linear combinations of the atoms (see, for example, MOD [38], K-*SVD* [32], and ℓ_1-regularized [39]).

Recently, a new algorithm named Structured Sparse Dictionaries for Sparse Representation (SRSSD) was presented in [40] and applied to a cognitive science study [18]; its fundamental characteristics is to search for sparse atoms and sparse coding simultaneously, reconciling, in this way, the two different dictionary-learning approaches.

Within the SRSSD method, we denote $\mathbf{X} \in \mathbb{R}^{n\times p}$ as a matrix where rows correspond to experimental observations (e.g., sequences of x and y coordinates that compose a trajectory, e.g., x1, y1, x2, y2, etc.). Note that n is the number of patches and p their length. Furthermore, we denote $\mathbf{V} \in \mathbb{R}^{p\times r}$ as a *dictionary*, whose r columns $\mathbf{V}^k$ represent the atoms learned by the dictionary. Thus, r is the number of atoms and p their maximum length. Note that given its sparsity, the SRSSD method can learn dictionary elements having different length, whose maximum we set to p (e.g., the number of nonzero consecutive elements can be $< p$). Finally, we define $\mathbf{U} \in \mathbb{R}^{n\times r}$ as the coefficient matrix.

The aim of SRSSD is finding out the best approximation of $\mathbf{X}$ in terms of $\mathbf{V}$ and $\mathbf{U}$ (i.e., $\mathbf{X} \approx \mathbf{U}\mathbf{V}^T$).

This problem can be formulated in terms of a minimization problem as follows:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2np}\|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2 + \lambda \sum_{k=1}^{r} \Omega_v(\mathbf{V}^k) \qquad \text{s.t. } \forall j, \Omega_u(\mathbf{U}_j) \leqslant \eta,\ \forall k\ \|\mathbf{V}^k\|_2 = 1 \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm for matrices, $\mathbf{U}_j$ are the rows of $\mathbf{U}$, $\Omega_v(\mathbf{V}^k)$ and $\Omega_u(\mathbf{U}_j)$ are norms or quasi-norms constraining (*regularizing*) the solutions of the minimization problem, with the parameters $\lambda \geqslant 0$ and $\eta \geqslant 0$ that control to which extension the dictionary and the coefficients are regularized, respectively. If one assumes that both Ω_u and Ω_v are convex, the problem (1) is convex w.r.t. $\mathbf{U}$ for $\mathbf{V}$ fixed, and vice versa.

Following [37], the structured sparsity of the atomsis imposed by setting

$$\Omega_v(\mathbf{V}^k) = \left\{ \sum_{i=1}^{s} \| \mathbf{d}^{G_i} \circ \mathbf{V}^k \|_2^{\alpha} \right\}^{\frac{1}{\alpha}} \tag{2}$$

where $\alpha \in (0,1)$, and each $\mathbf{d}^{G_i}$ is a p-dimensional vector satisfying the condition $d_j^i \geqslant 0$, with $G_i \in \mathcal{G}$ where $\mathcal{G}$ is a subset of the power set of $\{1,\ldots,p\}$, such that $|\mathcal{G}| = s$ and $\bigcup_{i=1}^{s} G_i = \{1,\ldots,p\}$. Thus, the vectors $\mathbf{d}^i$ define the structure of the atoms. More specifically, each $\mathbf{d}^{G_i}$ individuates a group of variables such that $d_j^i > 0$ if $j \in G_i$ and $d_j^i = 0$ otherwise. The norm $\| \mathbf{d}^{G_i} \circ \mathbf{V}^k \|_2^{\alpha}$ penalizes the variables selected by $d_j^i > 0$, hence, by this norm, each vector $\mathbf{d}^{G_i}$ induces non zero values for the j-th elements of the atom $\mathbf{V}^k$, when $j \in G$.

The resulting set of selected variables depends on the contribution of each $\mathbf{d}^{G_i}$ as described in [41]. For example, if the vectors $\mathbf{d}^{G_i}$ represent a partition on the set $\{1, \ldots, p\}$, then the penalization term (2) favours atoms $\mathbf{V}^k$ composed of non-zero variables belonging to just one part of the partition and so on: for specific choices of $\{G_i\}_{i=1}^s$, Ω_v leads to standard sparsity-inducing norms.

Nevertheless, the norm Ω_v expressed in Equation (2) is not differentiable and, consequently Equation (1) is not convex with respect to $\mathbf{V}$ for $\mathbf{U}$ fixed — although the converse is still true. By using results presented in [37, 42], we can write down Ω_v as a quadratic expression and reformulate the Equation (1) as:

$$\min_{\mathbf{U},\mathbf{V},\mathbf{H}} \frac{1}{2np}\|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2 + \frac{\lambda}{2}\sum_{k=1}^{r}\left[(\mathbf{V}^k)^T \mathrm{Diag}(\mathbf{Z}^k)^{-1}\mathbf{V}^k) + \|\mathbf{H}_k\|_\beta\right] \quad \text{s.t. } \forall j, \Omega_u(\mathbf{U}_j) \leqslant \eta, \quad \forall i \ \|\mathbf{V}^i\|_2 = 1 \tag{3}$$

where $\mathbf{H} \in \mathbb{R}_+^{r\times s}$ is a matrix satisfying the condition $h_{ki} \geqslant 0$ with $\beta = \frac{\alpha}{2-\alpha}$ and that minimizes the second term of the previous expression in which $\mathbf{Z} \in \mathbb{R}^{p\times r}$ has got its own elements defined as $z_{jk} = \left\{\sum_{i=1}^s \left(d_j^i\right)^2 (h_{ki})^{-1}\right\}^{-1}$. Notice that, as shown in [37], for both $\mathbf{U}$ and $\mathbf{V}$ fixed, the minimizer of (3) can be given in the closed form $\bar{h}_{ki} = |y_i^k|^{2-\alpha}\|\mathbf{y}^k\|_\alpha^{\alpha-1}$, for $k = 1, 2, ..., r$ and $i = 1, 2, ..., s$, where each $\mathbf{y}^k \in \mathbb{R}^{1\times s}$ is the vector $\mathbf{y}^k = (\|\mathbf{d}^1 \circ \mathbf{V}^k\|_2, \|\mathbf{d}^2 \circ \mathbf{V}^k\|_2, \ldots, \|\mathbf{d}^s \circ \mathbf{V}^k\|_2)$. Ultimately, we impose the constraint $\|\mathbf{V}^i\|_2 = 1$ to avoid solutions where $\mathbf{V}$ goes towards $\mathbf{o}$, or $\mathbf{U}$ becomes a sparse matrix, whose non-zero elements have excessively high values.

Since the functional in (3) is separately convex in each variable, we solve the minimization problem following the usual approach of alternating optimizations with respect to the values $\mathbf{H}$, to the coefficients $\mathbf{U}$ and to the dictionary $\mathbf{V}$. Most methods are based on this alternating scheme of optimization [43, 39] and its convergence towards a critical point of the functional is guaranteed by Corollary 2 of [44].

Algorithm 1 summarises the procedural steps of SRSSD, which comprises three stages:

S1 ***Matrix* H *update.*** In the first stage, both $\mathbf{U}$ and $\mathbf{V}$ are fixed and $\mathbf{H}$ values are updated. As said above, one can update $\mathbf{H}$ by computing $\bar{h}_{ki} = |y_i^k|^{2-\alpha}\|\mathbf{y}^k\|_\alpha^{\alpha-1}$. However, to avoid numerical instabilities near zero, we adopt the following smoothed update: $\mathbf{H}_k \leftarrow \max\{\bar{\mathbf{H}}_k, \epsilon\}$ with $\epsilon \ll 1$.

S2 ***Sparse Coding.*** In the second stage, both $\mathbf{V}$ and $\mathbf{H}$ are fixed and $\mathbf{U}$ values are updated. Note that equation (3) comprises two terms to be minimized, but the second term does not depend on $\mathbf{U}$. This implies that the optimization problem of equation (3) can be reformulated as: $\min_{\mathbf{U}} \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2$ *s.t.* $\forall j, \|\mathbf{U}_j\|_0 \leqslant \eta$. There are several well-known "pursuit algorithms" that find approximate solutions to this kind of problems, such as Basis Pursuit (BP) [45] and Orthogonal Matching Pursuit (OMP) [46]. Here, we approximate the ℓ_0 norm (a non convex problem) with its best convex approximation, i.e., ℓ_1 norm – which allows us to perform sparse coding by applying an Iterative Soft-Thresholding (IST) [47]. Note that this stage is equivalent to the application of Algorithm 2, where we apply only sparse coding with a fixed dictionary.

S3 ***Structured Atoms.*** In the third stage, both $\mathbf{U}$ and $\mathbf{H}$ are fixed and the dictionary $\mathbf{V}$ is updated; furthermore, following [37], an atom structured sparse representation can be found. As both $\mathbf{U}$ and $\mathbf{H}$ are fixed, the problem of equation (3) can be reformulated as:

$$\min_{\mathbf{V}} \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \frac{\lambda np}{2}\sum_{k=1}^{r}(\mathbf{V}^k)^T \mathrm{Diag}(\mathbf{Z}^k)^{-1}\mathbf{V}^k) \quad \forall\, i, \|\mathbf{V}^i\|_2 = 1 \tag{4}$$

In this case, as both terms of Equation (4) are convex and differentiable terms with respect to $\mathbf{V}$, a closed-form solution for $\mathbf{V}$ can be found. However, a proximal method is considered to avoid p matrix inversions:

$$\begin{aligned}\mathbf{V}^k \leftarrow \Pi_\nu(\mathrm{Diag}(\mathbf{Z}^k)\mathrm{Diag}(\|\mathbf{U}^k\|_2^2\mathbf{Z}^k + np\lambda\mathbf{I})^{-1}\cdot \\ \cdot(\mathbf{X}^T\mathbf{U}^k - \mathbf{V}\mathbf{U}^T\mathbf{U}^k + \|\mathbf{U}^k\|_2^2\mathbf{V}^k))\end{aligned} \tag{5}$$

where $\Pi_\nu(w)$ is simply the Euclidean projection of w onto the unit ball, and the argument of Π_ν is obtained by composing a forward gradient descent step on the first term with the proximity operator of the second term of (4).

4.2 Method for reconstructing test trajectories and calculating reconstruction error (RMS)

To build the dataset to be reconstructed, we split the trajectories into training segments or patches $X \in \mathbb{R}^{n\times p}$, where n is the number of patches (10000 in our test set) and $p/2$ is the length in time of the patches. Each row of X includes a sequence of x and y coordinates that compose a patch, e.g., $x1$, $y1$, $x2$, $y2$, etc. We split trajectories lasting more than $p/2$ time steps into two or more rows of X. Then, we use SRSSD to learn a dictionary $V \in \mathbb{R}^{p\times r}$ of r atoms, having maximum length p. For the sake of comparison, we adopt the same procedure using PCA (where the matrix V represents r principal components; but note that PCA requires all element to be the same length, which we fix to p).

In general, once learned the Dictionary V, X can be reconstructed by computing $U \in \mathbb{R}^{n\times r}$, where U represents the coefficients of the atoms computed with sparse coding (see Algorithm 2), or the PCA coefficients $U = XV$. Using the coefficients, a new reconstruction matrix $X_{rec} = UV^T$ (with $X_{rec} \in \mathbb{R}^{n\times p}$) can be obtained, which can be compared with the original X, to calculate a reconstruction error RMS. However, this method would only measure how well the learned dictionary represents the learned trajectories, not how well it generalises.

For this, we use a more compelling methodology, inspired by the *missing pixel method*: we create "holes" in the original trajectories (by removing consecutive columns of X) and measure how well our method permits to reconstruct them. More specifically, we firstly build a *restricted* matrix $X' \in \mathbb{R}^{n\times p'}$, with $p' < p$, by removing $p - p'$ (consecutive) columns from X. Therefore, we build a *restricted* dictionary $V' \in \mathbb{R}^{p'\times r}$ (o r PCA components), by removing $p - p'$ rows from V. We then compute the matrix $U' \in \mathbb{R}^{n\times r}$, using X' and V' (instead of X and V as above). Because U' has the same matrix dimensions as U, it is possible to use the original matrix V to compute $X'_{rec} = U'V^T$, with $X'_{rec} \in \mathbb{R}^{n\times p}$ that has the same dimension of original matrix X, thus resulting in a "full" trajectory with reconstructed holes.

Finally, we calculate the reconstruction error RMS using a *Frobenius* norm $\|\cdot\|_F$, which considers the difference between a target trajectory $X \in \mathbb{R}^{n\times p}$ and X'_{rec}.

$$\mathrm{RMS} \equiv \left\|X - X'_{rec}\right\|_F = \sqrt{\sum_i^n \sum_j^p \left(x_{ij} - x'_{rec,ij}\right)^2} \tag{6}$$

We adopt the same (missing pixel) procedure for both SRSSD and PCA, for test and validation. Furthermore, to test the robustness of the method, we use the learned dictionaries to reconstruct different portions of trajectories, from 10% to 90% – the latter coming closer to reconstructing full trajectories.

Algorithm 1 *SRSDD*$(\mathbf{X}, r, \eta, \lambda, \mathtt{stop_criterion})$

Require: matrix $\mathbf{X}$, number r of elements in the dictionary,
sparse coding parameter η, sparse atoms parameter λ
a termination condition "*stop_criterion*"
Ensure: Dictionary $\mathbf{V}$, Coefficients $\mathbf{U}$

initialize random $\mathbf{V}$ and $\mathbf{U}$
while (**check** $\mathtt{stop_criterion}$ is not satisfied) **do**

stage S1 – *update* H
compute $\mathbf{y}^k = (\|\mathbf{d}^1 \circ \mathbf{V}^k\|_2, \|\mathbf{d}^2 \circ \mathbf{V}^k\|_2, \ldots, \|\mathbf{d}^s \circ \mathbf{V}^k\|_2)$ for $k = 1, \ldots, r$
compute $\bar{h}_{ki} = |y_i^k|$ for $k = 1, \ldots, r$ and $i = 1, \ldots, s$
compute $\mathbf{H}_k \leftarrow \max\{\bar{\mathbf{H}}_k, \epsilon\}$ with $\epsilon \ll 1$

stage S2 – *sparse coding*
update $\mathbf{U}$ minimizing $\min_{\mathbf{U}} \frac{1}{np}\|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \eta \sum_{j=1}^{n} \|\mathbf{U}_j\|_1$ by IST

stage S3 – *structured atoms*
compute $z_{jk} = \left\{ \sum_{i=1}^{s} \left(d_j^i\right)^2 (h_{ki})^{-1} \right\}^{-1}$ for $k = 1, \ldots, r$ and $i = 1, \ldots, s$
compute
$\mathbf{V}^k \leftarrow \mathrm{Diag}(\mathbf{Z}^k)\mathrm{Diag}(\|\mathbf{U}^k\|_2^2 \mathbf{Z}^k + \lambda \mathbf{I})^{-1}(\mathbf{X}^T\mathbf{U}^k - \mathbf{V}\mathbf{U}^T\mathbf{U}^k + \|\mathbf{U}^k\|_2^2 \mathbf{V}^k)$
normalize the columns of $\mathbf{V}$ to have unit norm

end while

Algorithm 2 *Sparse Coding*$(\mathbf{X}, \mathbf{V}, \eta, \mathtt{stop_criterion})$

Require: matrix $\mathbf{X}$, Dictionary $\mathbf{V}$,
sparse coding parameter η, a termination condition "*stop_criterion*"
Ensure: Coefficients $\mathbf{U}$

initialize random $\mathbf{U}$
while (**check** $\mathtt{stop_criterion}$ is not satisfied) **do**

update $\mathbf{U}$ minimizing $\min_{\mathbf{U}} \frac{1}{np}\|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \eta \sum_{j=1}^{n} \|\mathbf{U}_j\|_1$ by IST

end while

ACKNOWLEDGMENTS

This research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2 to GP) and the Human Frontier Science Program (grant no. RGY0088/2014 to GP, MVDM and CK). The GEFORCE Titan GPU card used for this research was donated by the NVIDIA Corp. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

REFERENCES

[1] Andrea D'Avella and Emilio Bizzi. Shared and specific muscle synergies in natural motor behaviors. *Proceedings of the National Academy of Sciences*, 102(8):3076–3081, 2005.

[2] Andrea d'Avella, Alessandro Portone, Laure Fernandez, and Francesco Lacquaniti. Control of fast-reaching movements by muscle synergy combinations. *The Journal of Neuroscience*, 26(30):7791–7810, 2006.

[3] F. A. Mussa-Ivaldi and E. Bizzi. Motor learning through the combination of primitives. *Philos Trans R Soc Lond B Biol Sci*, 355(1404):1755–1769, Dec 2000.

[4] T. Iberall, G. Bingham, and M. A. Arbib. Opposition space as a structuring concept for the analysis of skilled hand movements. *Experimental Brain Research Series*, 15:158–173, 1986.

[5] Marco Santello, Martha Flanders, and John F. Soechting. Postural hand synergies for tool use. *Journal of Neuroscience*, 18(23):10105–10115, 1998.

[6] C. R. Mason, J. E. Gomez, and T. J. Ebner. Hand synergies during reach-to-grasp. *J Neurophysiol*, 86(6):2896–910, 2001.

[7] Andrea d'Avella. Modularity for motor control and motor learning. In *Progress in Motor Control*, pages 3–19. Springer, 2016.

[8] André EX Brown, Eviatar I Yemini, Laura J Grundy, Tadas Jucikas, and William R Schafer. A dictionary of behavioral motifs reveals clusters of genes affecting caenorhabditis elegans locomotion. *Proceedings of the National Academy of Sciences*, 110(2):791–796, 2013.

[9] Gordon J Berman, Daniel M Choi, William Bialek, and Joshua W Shaevitz. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, 11(99):20140672, 2014.

[10] Tiago V Gehring, Gediminas Luksys, Carmen Sandi, and Eleni Vasilaki. Detailed classification of swimming paths in the morris water maze: multiple strategies within one trial. *Scientific reports*, 5:14562, 2015.

[11] Neil Schmitzer-Torbert and A David Redish. Development of path stereotypy in a single day in rats on a multiple-t maze. *Archives italiennes de biologie*, 140(4):295–301, 2002.

[12] Alexander B Wiltschko, Matthew J Johnson, Giuliano Iurilli, Ralph E Peterson, Jesse M Katon, Stan L Pashkovski, Victoria E Abraira, Ryan P Adams, and Sandeep Robert Datta. Mapping sub-second structure in mouse behavior. *Neuron*, 88(6):1121–1135, 2015.

[13] Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954, 2016.

[14] Jumpei Matsumoto, Hiroshi Nishimaru, Taketoshi Ono, and Hisao Nishijo. 3d-video-based computerized behavioral analysis for in vivo neuropharmacology and neurophysiology in rodents. In *In Vivo Neuropharmacology and Neurophysiology*, pages 89–105. Springer, 2017.

[15] Haozhe Shan and Peggy Mason. Multiscale dictionary of rat locomotion. *arXiv preprint arXiv:1707.03360*, 2017.

[16] Avgoustinos Vouros, Tiago V Gehring, Kinga Szydlowska, Artur Janusz, Zehai Tu, Mike Croucher, Katarzyna Lukasiuk, Witold Konopka, Carmen Sandi, and Eleni Vasilaki. A generalised framework for detailed classification of swimming paths inside the morris water maze. *Scientific reports*, 8(1):15089, 2018.

[17] Rodolphe Jenatton, Julien Mairal, Francis R Bach, and Guillaume R Obozinski. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 487–494, 2010.

[18] Roberto Prevete, Francesco Donnarumma, Andrea D'Avella, and Giovanni Pezzulo. Evidence for sparse synergies in grasping actions. *Scientific Reports*, 8:616, 2018.

[19] G Tessitore and R Prevete. Designing structured sparse dictionaries for sparse representation modeling. In *Computer Recognition Systems 4*, pages 157–166. Springer, 2011.

[20] DR Amancio, ON Oliveira, and L da F Costa. On the concepts of complex networks to quantify the difficulty in finding the way out of labyrinths. *Physica A: Statistical Mechanics and its Applications*, 390(23):4673–4683, 2011.

[21] Maya Saar, Tomer Gilad, Tal Kilon-Kallner, Adar Rosenfeld, Aziz Subach, and Inon Scharf. The interplay between maze complexity, colony size, learning and memory in ants while solving a maze: A test at the colony level. *PloS one*, 12(8):e0183753, 2017.

[22] Anthony J Bagnall and Zhanna V Zatuchna. On the classification of maze problems. In *Foundations of Learning Classifier Systems*, pages 305–316. Springer, 2005.

[23] Edmund T Rolls, Simon M Stringer, and Thomas Elliot. Entorhinal cortex grid cells can map to hippocampal place cells by competitive learning. *Network: Computation in Neural Systems*, 17(4):447–465, 2006.

[24] Bailu Si and Alessandro Treves. The role of competitive learning in the generation of dg fields from ec inputs. *Cognitive neurodynamics*, 3(2):177–187, 2009.

[25] Jadin C Jackson, Adam Johnson, and A David Redish. Hippocampal sharp waves and reactivation during awake states depend on repeated sequential experience. *Journal of Neuroscience*, 26(48):12415–12426, 2006.

[26] Matthijs AA van der Meer, James J Knierim, D Yoganarasimha, Emma R Wood, and Mark CW van Rossum. Anticipation in the rodent head direction system can be explained by an interaction of head movements and vestibular firing properties. *Journal of neurophysiology*, 98(4):1883–1897, 2007.

[27] Adam Johnson, AndreA Fenton, Cliff Kentros, and A. David Redish. Looking for cognition in the structure within the noise. *Trends Cogn Sci*, 13(2):55–64, Feb 2009.

[28] T Hastie, R Tibshirani, and JH Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, corrected edition, 2003.

[29] H Zou, T Hastie, and R Tibshirani. Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics*, 15, 2004.

[30] B. Krishnapuram, L. Carin, M. A. T. Figueiredo, and A. J. Hartemink. Sparse multinomial logistic regression: fast algorithms and generalization bounds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(6):957–968, June 2005.

[31] M. Elad and M. Aharon. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, December 2006.

[32] M Aharon, M Elad, and A Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[33] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[34] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. *CoRR*, abs/0809.3083, 2008.

[35] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online Learning for Matrix Factorization and Sparse Coding. *Journal of Machine Learning Research*, 11(1):19–60, 2010.

[36] H. Shen and J. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6):1015–1034, July 2008.

[37] R Jenatton, G Obozinski, and F Bach. Structured sparse principal component analysis. *International Conference on AISTATS*, 2010.

[38] K Engan, SO Aase, and J Hakon Husoy. Method of optimal directions for frame design. In *Proceedings of ICASSP '99*, volume 5, pages 2443–2446. IEEE Computer Society, 1999.

[39] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems 19*, pages 801–808. Massachusetts Institute of Technology Press, 2007.

[40] G. Tessitore and R. Prevete. Designing structured sparse dictionaries for sparse representation modeling. In Robert Burduk, Marek Kurzynski, Michal Wozniak, and Andrzej Zolnierek, editors, *Computer Recognition Systems 4*, volume 95 of *Advances in Intelligent and Soft Computing*, pages 157–166. Springer Berlin / Heidelberg, 2011.

[41] R Jenatton, JY Audibert, and F Bach. Structured variable selection with sparsity-inducing norms. Technical report, arXiv:0904.3523, 2009.

[42] CA Micchelli and M Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

[43] C Basso, M Santoro, A Verri, and S Villa. Paddle: Proximal algorithm for dual dictionaries learning. *CoRR*, abs/1011.3728, 2010.

[44] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26:127–136, 2000.

[45] SS Chen, DL Donoho, and MA Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[46] JA Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory*, 50:2231–2242, 2004.

[47] I Daubechies, M Defrise, and C De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

SUPPLEMENTARY MATERIAL

Maze complexity measures

The figures shows below illustrate the results of maze discretization (binning), as used to measure maze spatial complexity (Figure 12); occupancy of animal R01 in the third phase of each day, using this binned space (Figure 13), the time required to complete a trial and secure a reward (Figure 14), and example trajectories during phase 3 of day 1 (Figure 15).

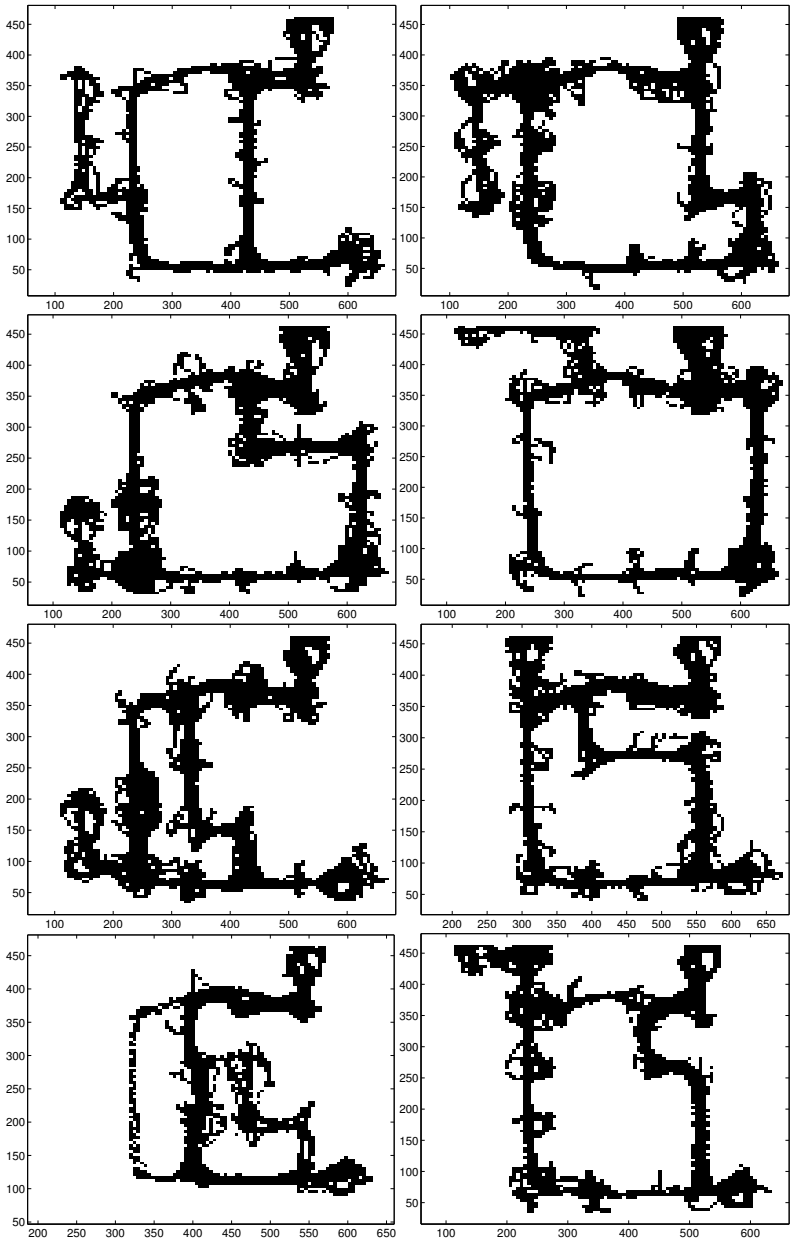


Figure 12: Discretization of mazes into bins used to measure maze spatial complexity

Experimental results, animals R02 and R03

In the main text, we have focused on animal R01 for illustrative purposes. Our results generalize to the two other two animals (R02 and R03) in our dataset. Figures 16 and 17 shown below illustrate that 1) reconstruction error is lower using SRSSD

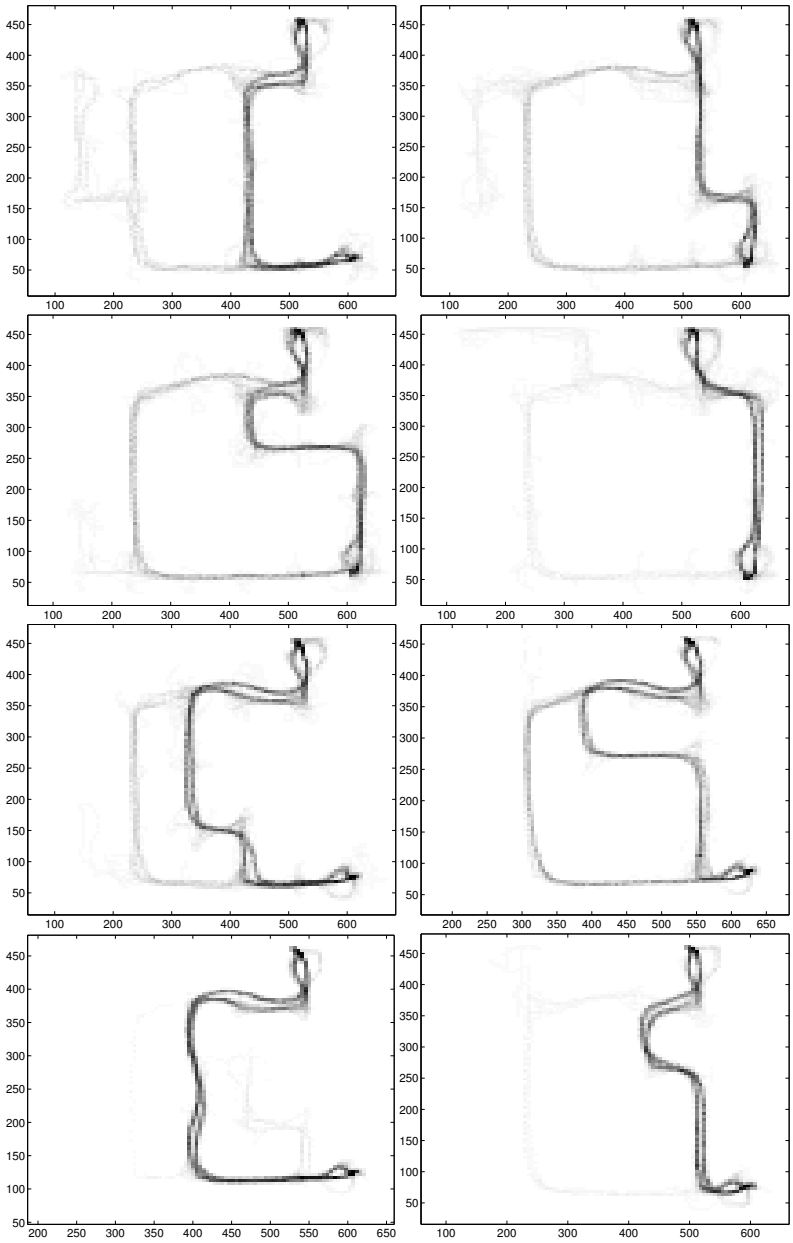


Figure 13: Occupancy of R01 in the third phase of each day, after binning the space.

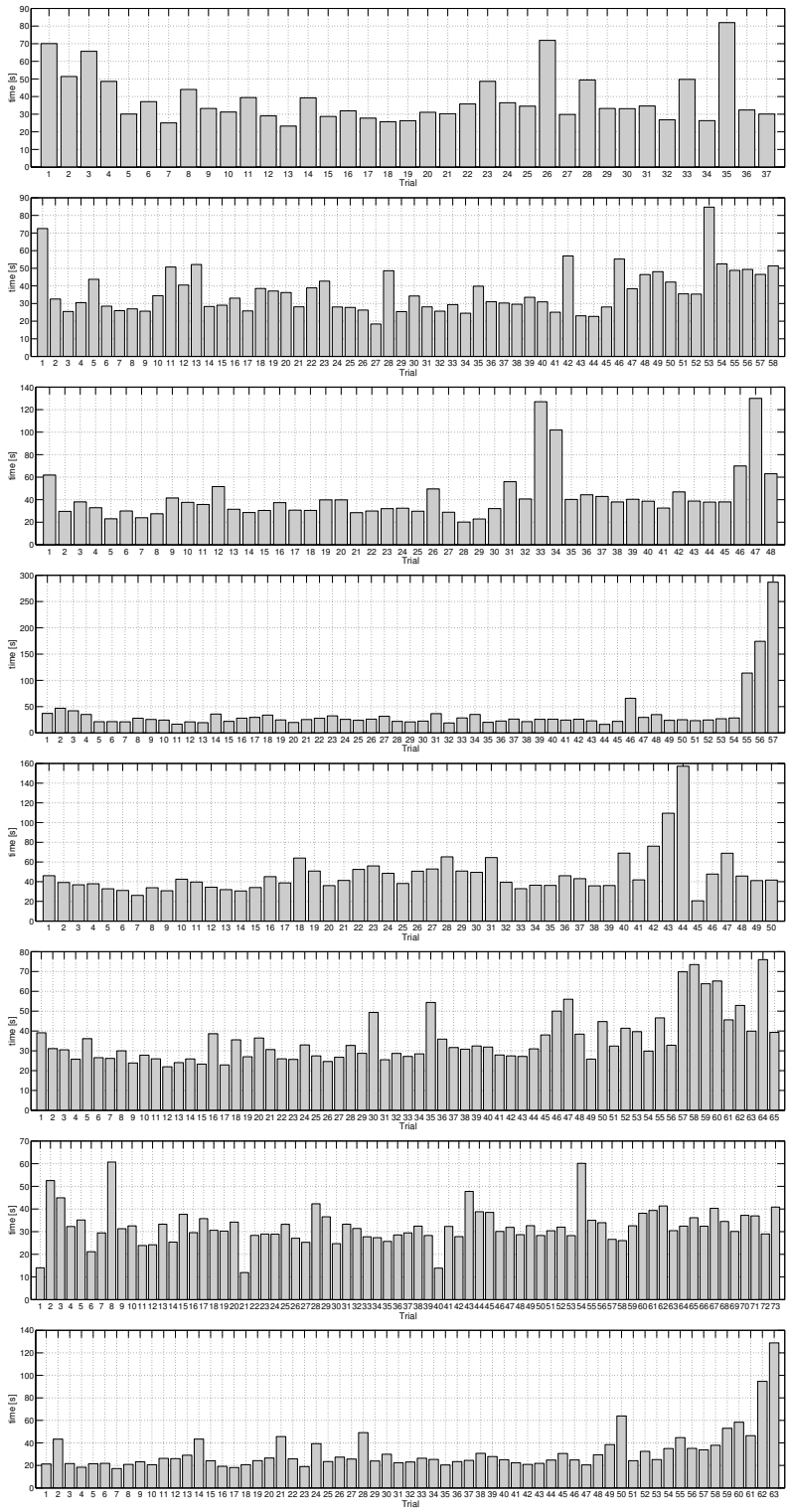


Figure 14: Time required for completing a trial (and collecting a reward) for each day, animal R01

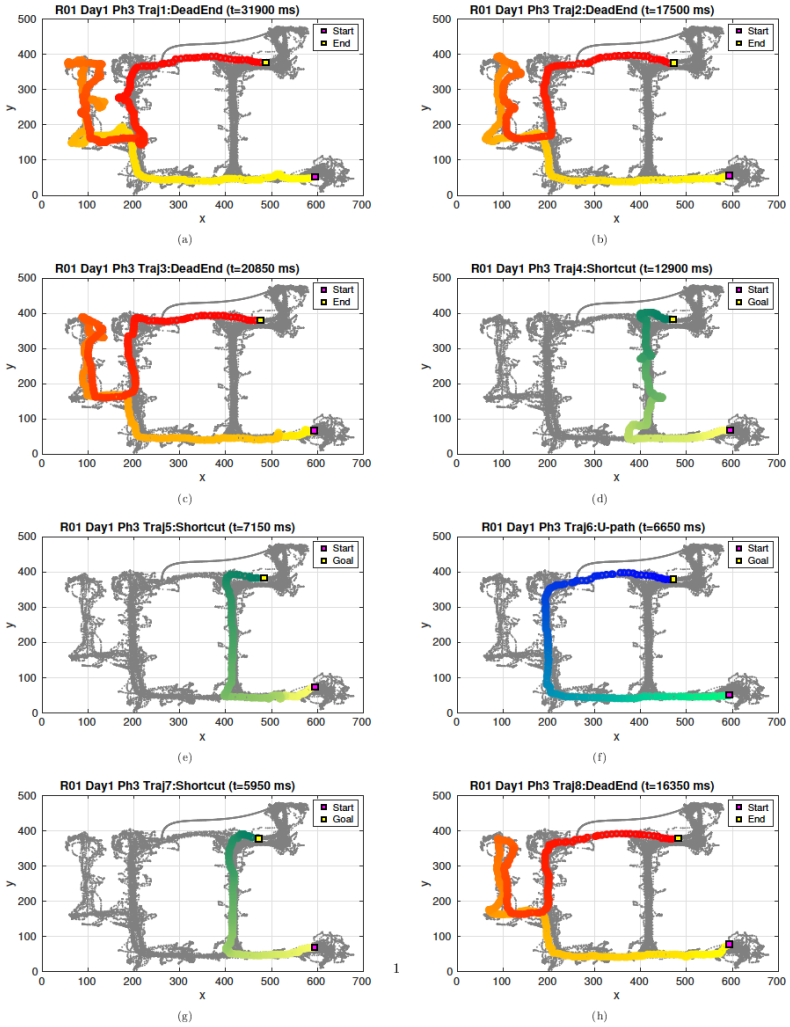


Figure 15: Eight examples of trajectories classified as U-path, shortcut or dead-end, during day 1, phase 3, in animal R01

compared to PCA in both animals R02 and R03; 2) in both animals, coefficient sparsity correlates with measures of spatial complexity of the mazes (compare to Figure 7).

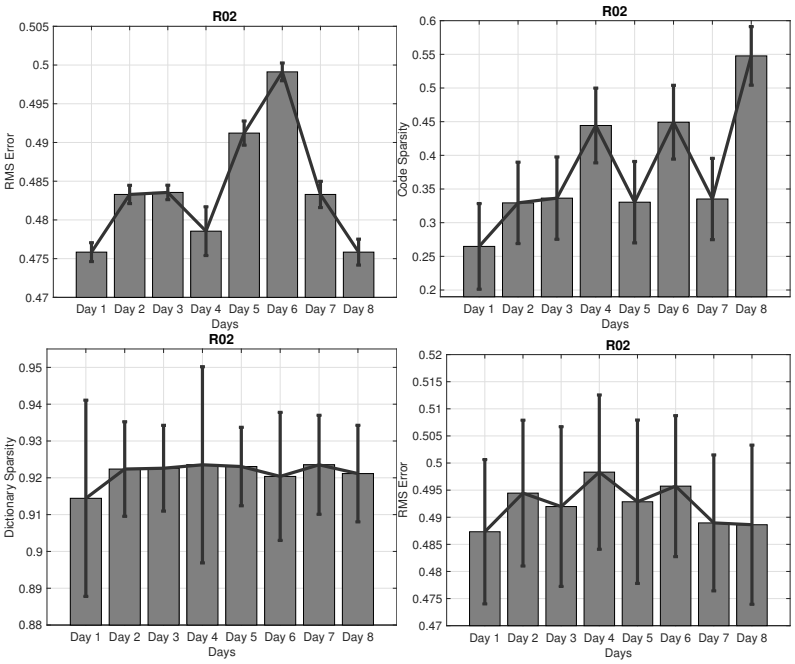


Figure 16: Reconstruction across 8 days of R02 (after learning only from data of day 1. Noise is 0.5 and the best 30 SRSSD dictionaries found on day 1 are used. (a): RMS Error. (b): coefficient sparsity. (c) dictionary sparsity. (d) Baseline performance: reconstruction of Phase 1.

Figure 18 shows that coefficient sparsity of animals R02 and R03 increases in phase 1 of day 1, while the animals are learning of the U maze, akin to R01 (discussed in the main text). Figures 20 and 20 show that classification of (U-path or shortcut) trajectories is accurate for R02 and R03, respectively, akin to R01 (discussed in the main text).

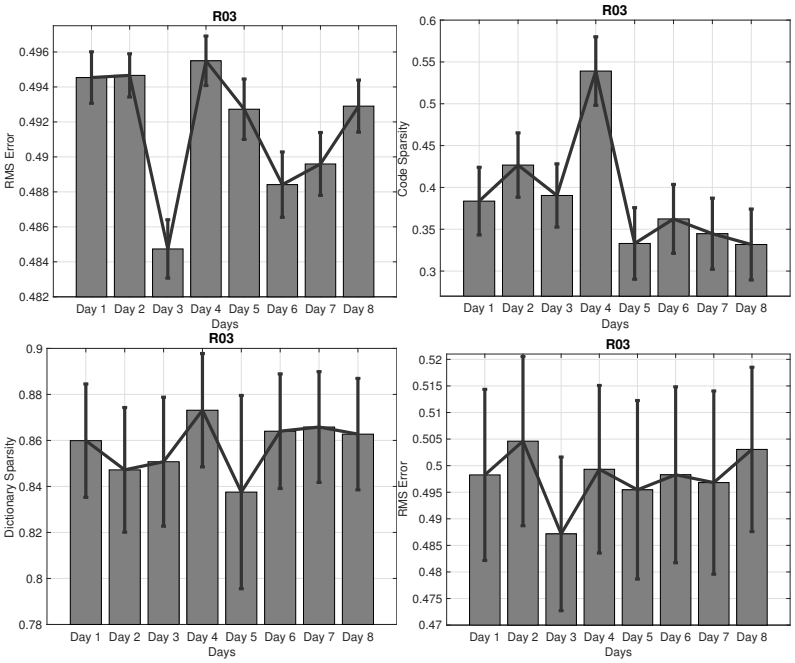
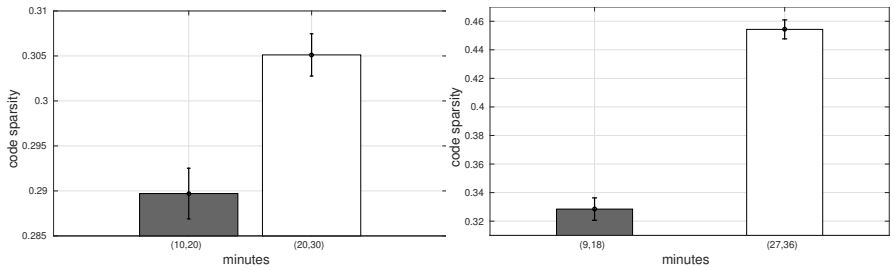


Figure 17: Reconstruction across 8 days of R03 (after learning only from data of day 1. Noise is 0.5 and the best 30 SRSSD dictionaries found on day 1 are used. (a): RMS Error. (b): coefficient sparsity. (c) dictionary sparsity. (d) Baseline performance: reconstruction of Phase 1.



(a) Coefficient sparsity, R02 **(b)** Coefficient sparsity, R03

Figure 18: Coefficient sparsity of animals R02 and R03 increases in phase 1 of day 1, while the animals learn navigating the U maze, thus revealing stereotypy of behaviour. We report the comparison between the 2nd and 4th intervals of the same length.

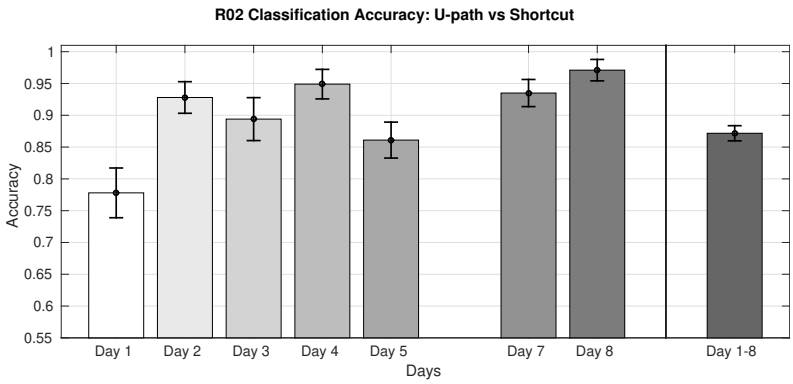


Figure 19: Classification of trajectories (U-path or shortcut) of phase 3, day 1, animal R02. The score for day 6 is missing, since on that they R02 always selects the shortcut.

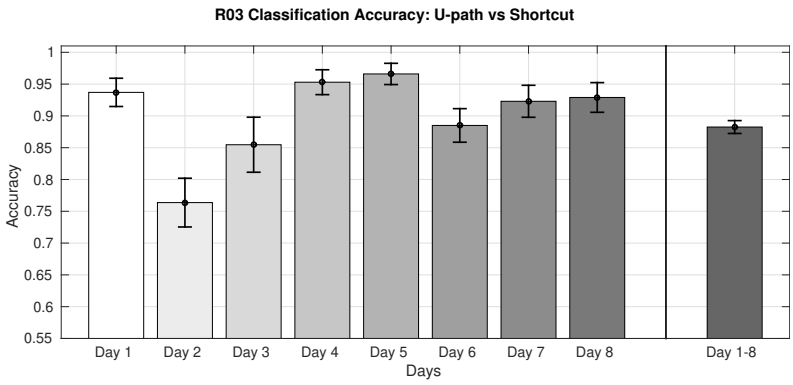


Figure 20: Classification of trajectories (U-path or shortcut) of phase 3, day 1, animal R03