Journal of Neuroscience Methods 300 (2018) 184–195

Contents lists available at ScienceDirect

Journal of Neuroscience Methods

journal homepage: www.elsevier.com/locate/jneumeth

Automated gesture tracking in head-fixed mice

A. Giovannucci[a,b,*], E.A. Pnevmatikakis[a], B. Deverett[b,c], T. Pereira[b], J. Fondriest[b,c], M.J. Brady[b,c], S.S.-H. Wang[b,c], W. Abbas[d], P. Parés[d], D. Masip[d]

[a] Center for Computational Biology, Flatiron Institute, Simons Foundation, New York, NY, USA
[b] Princeton Neuroscience Institute and Department of Molecular Biology, Princeton University, Princeton, NJ, USA
[c] Robert Wood Johnson Medical School, New Brunswick, NJ, USA
[d] Department of Computer Science, Universitat Oberta de Catalunya, Barcelona, Spain

HIGHLIGHTS

- The head-fixed mouse preparation offers unique advantages in a variety of experimental contexts.
- We present an affordable and flexible framework for measuring behavior in head-fixed preparations.
- We benchmark our algorithms against ground truth data and demonstrate good performance.
- The framework is a valuable complement to recording and stimulation technologies.

ARTICLE INFO

Article history:
Received 31 January 2017
Received in revised form 25 June 2017
Accepted 13 July 2017
Available online 17 July 2017

MSC:
00-01
99-00

Keywords:
Behavior
Head-fixed
Tracking

ABSTRACT

Background: The preparation consisting of a head-fixed mouse on a spherical or cylindrical treadmill offers unique advantages in a variety of experimental contexts. Head fixation provides the mechanical stability necessary for optical and electrophysiological recordings and stimulation. Additionally, it can be combined with virtual environments such as T-mazes, enabling these types of recording during diverse behaviors.
New method: In this paper we present a low-cost, easy-to-build acquisition system, along with scalable computational methods to quantitatively measure behavior (locomotion and paws, whiskers, and tail motion patterns) in head-fixed mice locomoting on cylindrical or spherical treadmills.
Existing methods: Several custom supervised and unsupervised methods have been developed for measuring behavior in mice. However, to date there is no low-cost, turn-key, general-purpose, and scalable system for acquiring and quantifying behavior in mice.
Results: We benchmark our algorithms against ground truth data generated either by manual labeling or by simpler methods of feature extraction. We demonstrate that our algorithms achieve good performance, both in supervised and unsupervised settings.
Conclusions: We present a low-cost suite of tools for behavioral quantification, which serve as valuable complements to recording and stimulation technologies being developed for the head-fixed mouse preparation.

1. Introduction

The preparation consisting of a head-fixed mouse running on a spherical or cylindrical treadmill (Fig. 1(a)) has been used in a variety of experimental contexts, including electrophysiological and optical neural recordings (Dombeck et al., 2007; Najafi et al., 2014; Heiney et al., 2014b), neuronal stimulation (Heiney et al., 2014a; Lee et al., 2015; Witter et al., 2013; Guo et al., 2015), behavioral studies (Chettih et al., 2011; Kloth et al., 2015; Lee et al., 2015; Hira et al., 2015; Heiney et al., 2014b), and virtual reality setups (Harvey et al., 2009; Dombeck et al., 2009; Sofroniew et al., 2014). Although preparations with freely moving animals are ideal for the study of social interaction and ecologically meaningful movements, they lack the brain stability necessary for recordings and manipulations through cranial windows. Head fixation solves this problem and can be easily combined with environments that allow mice to

* Corresponding author at: Center for Computational Biology, Flatiron Institute, Simons Foundation, New York, NY, USA.
E-mail address: agiovannucci@flatironinstitute.org (A. Giovannucci).

https://doi.org/10.1016/j.jneumeth.2017.07.014

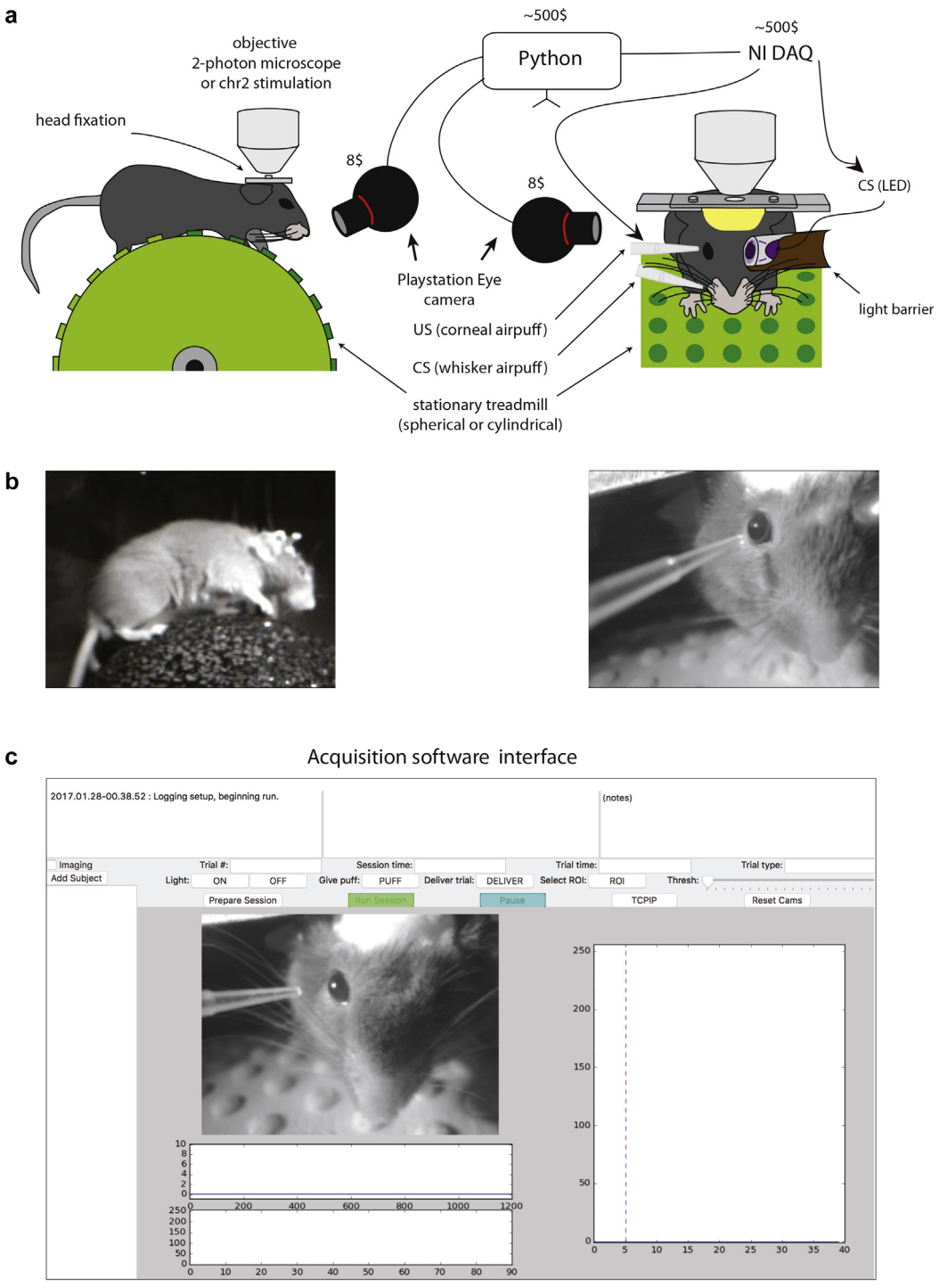

Fig. 1. Experimental and acquisition setup. (a) Schematic representation of the head-fixed preparation. The mouse is free to move on a cylindrical or spherical treadmill with minimal friction. At the same time the brain can be imaged via one or two-photon calcium imaging or optically stimulated via focused lasers or LEDs. Up to two PS3 eye cameras can be positioned in front or/and on the right side of the mouse and connected to the software. Other equipment for stimulation (like air puffers or eye directed LED light) can be positioned in front or on the side of the mouse. (b) Still image of the mouse in the two configurations (left and right). A light barrier can be added on the eye of the mouse to deliver light stimuli and not interfere with the optical recordings. (c) Interface of the open source software implemented for the acquisition of multi-camera data.

learn complex behavioral tasks (Chettih et al., 2011; Harvey et al., 2009; Sofroniew et al., 2014).

In head-fixed preparations, quantitative descriptions of behavior can be obtained by manual annotation (Dombeck et al., 2007), physiological measurements such as EMG (Hira et al., 2009; Chettih et al., 2011), or using a video camera (Chettih et al., 2011). A plethora of methods have been developed for extraction of movement parameters from video recordings, many of which rely upon carefully tuned features that generalize poorly across experimental setups (Heiney et al., 2014a,b; Witter et al., 2013; Hira et al., 2015; Schneider et al., 2014; Lee et al., 2015; Clack et al., 2012; Guo et al., 2015; Hoogl et al., 2015). Neuroscientists currently lack a turn-key, low-cost, general-purpose and scalable framework for automatic behavioral quantification in head-fixed mice.

Our aim in this paper is to present a flexible and user-friendly behavioral acquisition and analysis system that leverages the advantages of the head-fixed preparation, and to evaluate its performance in measuring behavior in comparison to manually annotated data. Our system consists of software for the acquisition of high-speed videos from multiple cameras, along with scalable algorithms for quantifying mouse behavior, with emphasis on measuring locomotion speed and motion patterns of whiskers, paws and tail. The proposed video acquisition system enables recording speeds above 150 frames per second from multiple cameras, and it is inexpensive in terms of both hardware (less than 1000$) and software (Python, open source). Additionally, it allows versatile stimulus delivery in closed loop with behavioral analysis. Further, we provide two algorithms to quantify mouse behavior that require minimal user intervention, and that scale well with large data sets: Flow-Loc and Block-Loc.

Flow-Loc is a flexible, unsupervised approach that quantifies spatially localized and stereotyped movements, i.e., movements that repeatedly span the same area within the field of view, e.g., whisker or whisker pad, eyelid, nose, or analogous body displacements in head-fixed rodents. The algorithm first estimates the frame-by-frame local motion fields in the movie using dense optical flow algorithms (Farnebäck, 2003), and then seeks a low dimensional representation of such fields using matrix factorization techniques with side constraints. The algorithm automatically identifies patches of pixels undergoing coherent motion patterns and partially demixes patches with overlapping motion fields. Although Flow-Loc can identify local motion fields, it fails to capture long-range movements, such as a paw traveling across the full field of view. Moreover, it cannot be applied to closed loop experiments because of computational limitations. To complement it, we introduce Block-Loc, a fast supervised algorithm which can track in real time movements spanning large areas. Block-Loc is trained to rapidly identify rectangular blocks of pixels enclosing specific animal features (such as the limbs or tail) by efficiently implementing classifiers operating on sliding windows.

The manuscript is organized as follows. In Section 2 we provide technical details about the experimental methods, the software and algorithms proposed in the manuscript, and the empirical evaluation of the presented algorithms. In Section 3 we outline our contributions in terms of software and algorithms, and provide an empirical evaluation of their performance in terms of accuracy and computing time. Finally, in Section 4 we summarize our contributions with respect to current state-of-the-art methods and outline the implications of our work for the neuroscience community.

2. Methods

2.1. Experimental procedures

Experimental procedures were carried out as approved by the Princeton University Institutional Animal Care and Use Committee and performed in accordance with the animal welfare guidelines of the National Institutes of Health. The same preparations we employed in our behavioral analysis were also used for some imaging experiments. The animal preparation was performed as in Najafi et al. (2014). We used 3 male 12- to 16-week-old C57BL/6J mice (Jackson Laboratory), housed in reversed light cycle. Mice underwent anesthesia (isoflurane, 5% induction, 1.0–2.5% maintenance) and a 3-mm-wide craniotomy was drilled over the paranormal area of cerebellar lobule VI. A custom-made two-piece headplate (Dombeck et al., 2009) was attached to the animal's head. After 15 h delay for animal recovery, the top plate was removed for delivery of 400–800 nL of virus, (∼50 nL/min, AAV1.Syn.GCaMP6f.WPRE.SV40 [Penn Vector Core, lot AV-1-PV2822]) in two injections ∼750 and ∼1250 μm lateral to the midline and 250 μm deep from the dura surface using borosilicate glass pipettes (World Precision Instruments, 1B100F-4, 1/0.58 mm OD/ID). All animals were placed back in their home cage for 2 weeks of recovery.

Animals were first habituated to a cylindrical or spherical treadmill that rotates along a single axis (Fig. 1(a)) for repeated intervals over 5 days of incremental exposure. After habituation, animals were exposed to a variety of stimuli, sometimes while simultaneously measuring brain activity under a two-photon microscope. Some of the movies are taken from animals that were undergoing eyeblink conditioning (Kloth et al., 2015). Training consisted of repeated parings of two stimuli, either puff-puff or light-puff, separated by intervals of 250 or 440 ms respectively. This training often induced movements in an otherwise-still mouse. The stimuli consisted of (i) a periorbital airpuff (10–20 psi, 30 ms in duration, delivered via a plastic needle placed 5 mm from the cornea and pointed at it, (ii) a flash of light (400 nm, 500 ms), or (iii) an airpuff to whisker vibrissae (2–3 psi, 800 ms).

2.2. Hardware and software setup for image acquisition

The mechanical parts to establish the head-fixed setup with a cylindrical treadmill are inexpensive and include an Exervo TeraNova® EVA foam roller (about 25$ per 5 treadmills) plus various bearings and mechanical posts and joints (see Chettih et al., 2011 for details).

We acquired all the movies using the open-source acquisition software we present in this paper (https://github.com/bensondaled/eyeblink). The package is implemented in Python and allows recordings from multiple cameras (Fig. 1). In its most basic form, the setup requires one desktop computer and one webcam. In this paper, we employ PlayStation 3 Eye® camera, which can record a field of view of 320 × 240 pixels at speeds up to 187 Hz, thus enabling an inexpensive high-speed recording system. To use the PS3 camera in dark conditions, the infrared filter was removed by opening the case and carving it out with a screw driver. Illumination was provided by an array of IR LEDs. The PS3 was interfaced with the data acquisition computer using the CLEye driver (CodeLaboratories, https://codelaboratories.com/products/eye/) and SDK. The presented Python package can acquire images at frame rates ranging from 15 Hz to 125 Hz, and resolutions of 640 × 480 or 320 × 240 pixels for each camera. For most 2-camera preparations, each camera can acquire images at up to 60 Hz with a resolution of 320 × 240 pixels. If synchronization and/or stimulus delivery are required, then the additional data acquisition board NI-USB-8451 (less than 500$) may be included. Stimuli can be delivered using the same custom Python software package (Fig. 1(a)). Control signals are generated from within Python and sent to a National Instruments card (NI-USB-8451) using the PyDAQmx wrapper for the National Instruments DAQmx C API (http://pythonhosted.org/PyDAQmx/). For light delivery, digital outputs can be directly connected to LED drivers or laser controllers. For air puff delivery, a digital output can be connected to a circuit gating a solenoid valve.

2.3. Flow-Loc

Flow-Loc is an algorithm that estimates movements that repeatedly span the same localized regions of the FOV. This is achieved by combining dense optical flow algorithms (Farnebäck, 2003) and dictionary learning (DL) (Olshausen and Field, 1996).

Dense optical flow algorithms (Farnebäck, 2003) compare two contiguous frames (fr_i and fr_{i+1}) in order to find a vector-valued optical flow function $flow(\cdot,\cdot)$ for each frame such that:

$$fr_i(y,x) \simeq fr_{i+1}(y + flow(y,x)[0], x + flow(y,x)[1])$$

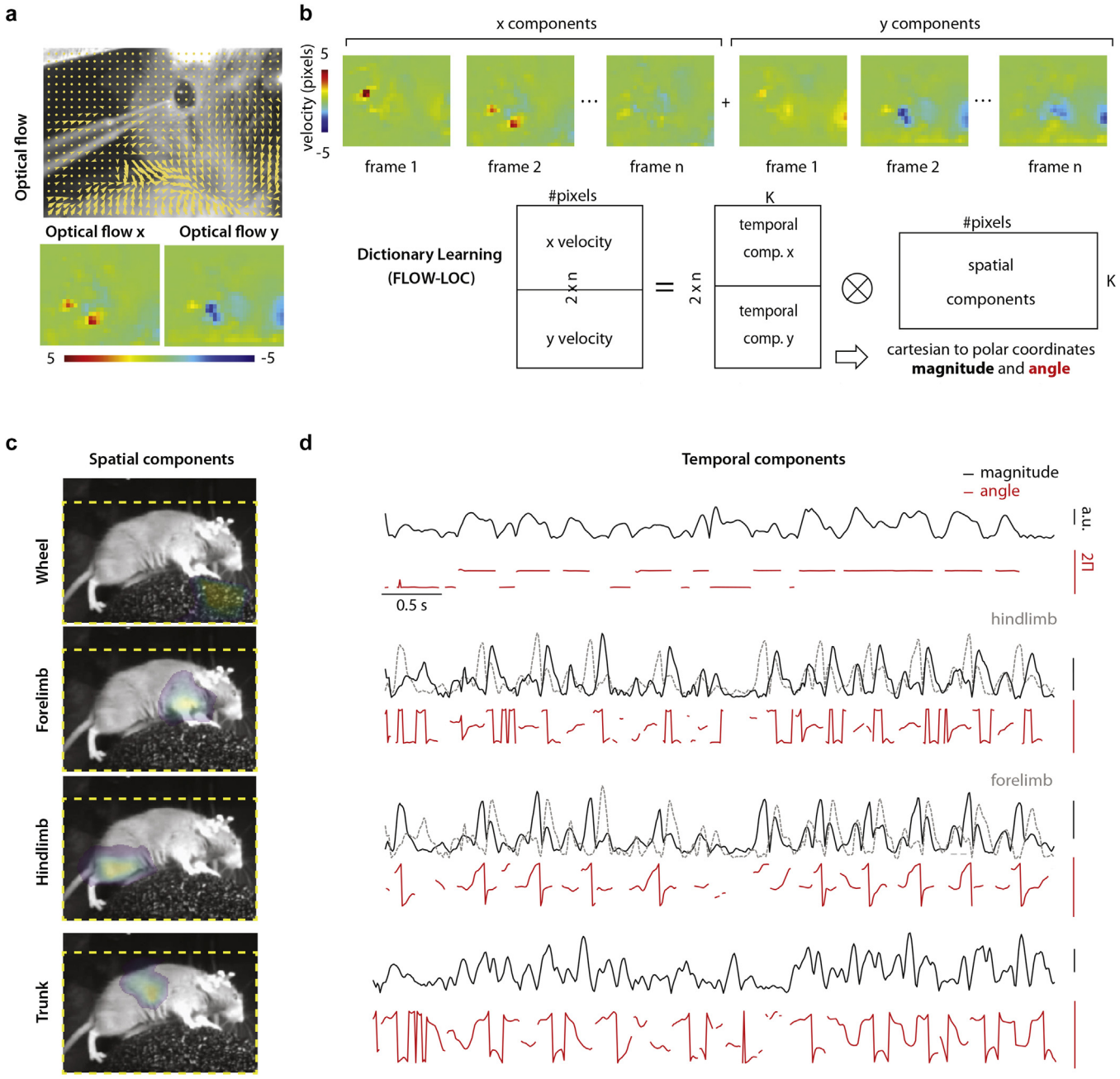

Fig. 2. Flow-Loc: fully automatic detection of movement. (a) Top: Still image of the mouse front camera with overlaid a subsampled version of the associated optical flow (yellow arrows). The length of the arrow represents the average velocity of a pixel's neighborhood between the current and the previous frame. Bottom: Heat map representation of the optical flow along *x* (left) and *y* (right). (b) Using dictionary learning to extract regions of interest and corresponding velocity trajectories. *Top: x* and *y* components of the velocity field are concatenated along the time dimension. The colormap represents the intensity of the vector field at each pixel. *Bottom:* The resulting 3D tensor is reshaped into a 2D matrix and then factorized via DL using K components. The temporal components (a concatenation of *x* and *y* velocity field temporal values), are mapped into polar coordinates. (c, d) Flow-Loc on front camera. (c) Still mouse image from lateral camera with overlaid the spatial components extracted by DL. The components are localized and corresponds to relevant regions of interest (from top to bottom wheel, forelimb, hindlimb, trunk). (d) Magnitude (black) and angle (red) in polar coordinates extracted by the temporal components and corresponding to the spatial components to the left. The angle is defined only when the magnitude is larger than half of its standard deviation (the angle is very noisy when the magnitude is small, see the methods for an explanation). For hindlimb and forelimb overlaid in dashed gray the movement of the rostro-caudally opposite limb to highlight the alternating pattern of the gait. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The algorithm relies on an efficient polynomial approximation of pixel neighborhoods to infer displacement fields. The output is a 2D vector field that assigns to each pixel the average inter-frame velocity of a parameterizable neighborhood (see Fig. 2(a) and extended data movie M.1). The velocity space inferred by the optical flow algorithm is a more convenient manifold than the pixel space for describing movement. In order to extract spatiotemporally coherent patterns from the velocity fields, we apply DL, a matrix factorization technique that, given a matrix *Y*, finds two matrices *A* (non-negative) and *B* such that:

$$\min_{A\geq 0,B}\|Y - AB\|_F^2. \quad (1)$$

In our case, the velocity field is composed of two tensors, denoted by $V_x, V_y \in \mathbb{R}^{r\times c\times t}$ where *r*, *c* and *t* are the number of rows, columns, and frames, respectively. Here, for each frame *i*, V_x and V_y correspond respectively to the matrices *flow*(*y*, *x*)[1] and *flow*(*y*,

x)[0], when applying the dense optical flow algorithm between the frames $i-1$ and i. To apply DL (see Fig. 2(b)) we first reshape V_x and V_y into 2D matrices, V_x, $V_y \in \mathbb{R}^{rc\times t}$, and concatenate them along the time axis to get a single matrix $V_{OF} \in \mathbb{R}^{rc\times 2t}$. The DL problem can then be cast as:

$$\begin{aligned} &\min_{S\geq 0,T} && \|V_{OF} - TS\|_F^2 \\ &\text{s.t.} && \forall i, \quad \|s_i\|_1 \leq \lambda \\ & && \forall j, \quad \|t_j\|_2^2 \leq 1 \end{aligned} \tag{2}$$

where $S \in \mathbb{R}^{rc\times K}$ and $T \in \mathbb{R}^{2t\times K}$ will encode space and time components respectively, and K is a user-defined parameter that sets the number of components (dictionary elements) that the algorithm will search for. The intuition behind this approach is that the movement of each local body part will translate into a set of pixels that have highly correlated values in the optical flow function. This motion can be captured by the outer product between a column **s** of the spatial matrix S, that describes the spatial extent of the pixel set, and a row **t** of the temporal matrix T that describes the velocity vector of the components in the x and y directions (Fig. 2(b)).

There are two main advantages of the DL approach over simpler approaches such as principle component analysis. The first is that DL restricts the spatial components to be non-negative, thus giving a physical interpretation to the S matrix, and the second is that it does not require orthogonality between its columns. The DL matrix tries to decompose the space-time motion pattern into a small number of components, making it an effective tool for dealing with overlapping structures. Moreover, it promotes sparse representations through the regularization parameter λ (Olshausen and Field, 1996). The matrix of velocity vectors T, on the other hand, can take both positive and negative values, corresponding to opposite movement directions, and thus it lacks the sign and sparsity constraints of the standard DL framework.

We can speed up the solution of the factorization problem by relying on online dictionary learning (Mairal et al., 2010). In this case the objective function can be decomposed as:

$$\begin{aligned} &\min_{s_i\geq 0,T} && \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|v_i - Ts_i\|_2^2 \\ &\text{s.t.} && \forall i, \|s_i\|_1 \leq \lambda \\ & && \forall j, \quad \|t_j\|_2^2 \leq 1, \end{aligned} \tag{3}$$

where index i iterates over the pixels, $v_i \in \mathbb{R}^{2t\times 1}$ is the motion of pixel i across time, and $s_i \in \mathbb{R}^{K\times 1}$ is a vector representing the weight of contribution of each temporal component to pixel i. The dictionary contains the temporal components, a sparse linear combination of which can describe any pixel. This approach has the advantage that it can be efficiently solved by online algorithms, reducing the demand on computational time and memory (Mairal et al., 2010).

Although Flow-Loc operates based on online sparse DL, it cannot currently operate on a live stream of frames, since it computes on a pixel-by-pixel basis as opposed to a frame-by-frame basis. This is due to the sparse representation of movement patterns, in which the activity of any pixel in the FOV is approximated from a sparse set of movements based upon pixel location (e.g., paw, whisker pad, etc.). Future work may address an extension of this algorithm to handle frame-by-frame analysis and closed loop applications.

In order to provide a more intuitive output, the Cartesian coordinates x and y can be transformed into polar coordinates (see Fig. 2(b) bottom). This allows easier visualization of movement metrics. When the movement magnitude is very small, the optical flow is a random fluctuation of values around zero subject to measurement noise. In the present plots and analyses, we define the angle only in frames where the magnitude is larger than a threshold computed from the signal variability (half of one standard deviation).

2.3.1. Choosing the right number of components

Choosing the number of components in matrix factorization is a challenging problem from a statistical point of view, and many methods have been proposed, including ones based upon cross-validation (Owen and Perry, 2009) or sparsity-based criteria (Bengio et al., 2009). In our case, we take advantage of the interpretation of each factor as a specific consistent movement in the animal's body. Once the algorithm is tuned to the input features and the measured variables, it generalizes well to similar scenarios. In our experiments, the criteria used to select the component cardinality are based on the number of clearly recognizable movements in the movie and the salience of the movement to be measured (i.e. wheel movement is much clearer than whisker movement). One should always include enough components to capture clearly visible movements, and more components should be included when the movement of interest is poorly visible or exhibits spatial overlap with other moving objects. For instance, if one runs the algorithm on a small region, as in Fig. 3(a), one component is sufficient to capture the movement of the only underlying object, the wheel. If one selects a similarly small region over the right whiskers, then at least two components will be needed, since the whiskers and wheel overlap in space. Finally, if one selects larger areas, as in 3 c, one component is required for each movement: wheel, left whiskers, right whiskers, nose, and so on.

Although Flow-Loc can estimate local velocity fields, recovering motion parameters with perfect accuracy is not always feasible, since the DL approach assumes that all the pixels in a spatial component (ROI) have the same velocity direction, a condition that is satisfied only if the movement is completely rigid. Instead, a movement density is extracted, whose time course is highly correlated (see Section 3.2.1) with behavioral variables, but whose magnitude might be not directly interpretable.

2.4. Block-Loc

Block-Loc is a supervised algorithm that localizes specific features of the mouse in video frames. It requires training with a moderate quantity of labeled samples, and detection operates at faster than real time speeds. The classification algorithm is composed of two steps (Fig. 4(a)–(c)): (i) Feature extraction using Haar filters on all the possible rectangular sub-windows present in the Region of Interest (sliding windows search); and (ii) efficient classification of each sub-window as limb/non-limb using a cascade of Adaboost classifiers.

We ran Block-Loc on two example portions of the mouse body: paws and base of tail. Using a manually annotated training set consisting of 4217 frames for paws and 1053 for tail, we computed the Haar-like features as described in Viola and Jones (2001). Haar filters convolve the image with rectangular functions, and have been extensively used in computer vision tasks such as face detection (Viola and Jones, 2001) and object recognition (Lienhart and Maydt, 2002). The computation of the Haar descriptors can be significantly sped up using the integral image. The integral image I provides an effective method for computing the sum of values within a rectangular sub-shape in an image. Each position $I(x, y)$ contains the sum of all the pixels above and to the left of the original image ($I(x, y) = \sum_{\forall i\leq x, j\leq y} Im(i, j)$). Fig. 4(b) illustrates an example of the computation of the sum of the pixels in the red region marked as D. The integral image I at point D contains the sum of all the pixels from the upper left corner to D. In order to compute the sum of pixels contained in the red area, one must subtract C and B from D, and add A (which was subtracted twice as a part of B and C). Convolutions with rectangular Haar filters

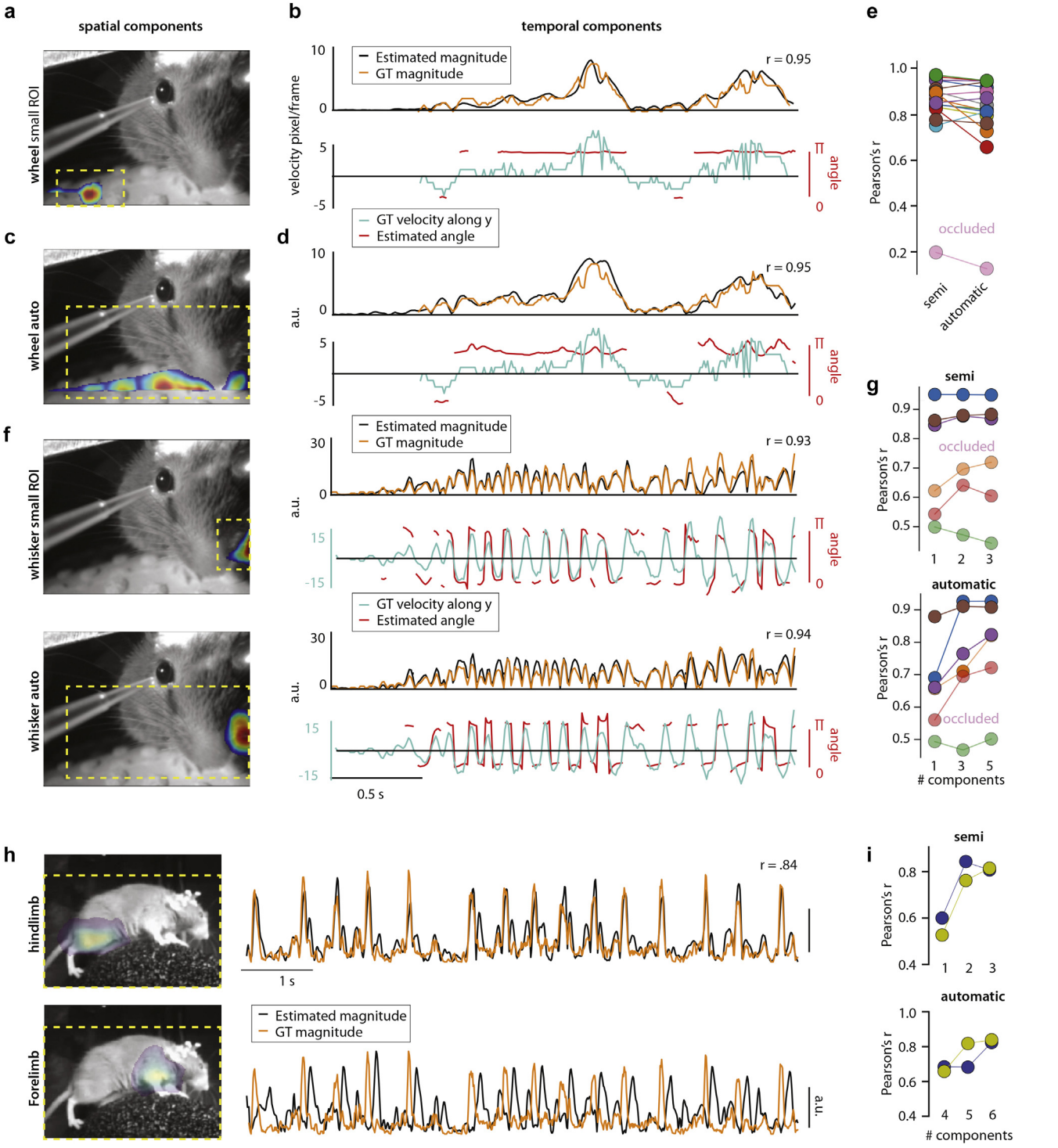

Fig. 3. Flow-Loc performance evaluation against ground truth. (a) Still image of the mouse overlaid to the best detected spatial component (heat map) and the crop (dashed yellow) used as input for semi-automatic (small ROI) movement inference. (b) Temporal components in polar coordinates corresponding to (a). *Top*: Temporal component estimated magnitude (black) overlaid to the ground truth (GT) velocity magnitude (orange). *r* is the Pearson's correlation coefficient between estimated and ground truth velocity magnitude. *Bottom*: Estimated direction (red, angle) overlaid to the ground truth velocity (cyan, with sign indicating treadmill rotation direction). The angle is not defined for each frame. (c, d) Same as (a, b) but for automatic (large ROI, yellow dash line in c) selection. (e) Correlation coefficient of estimated vs ground truth magnitude in the case of semiautomatic (left) and automatic (right) inference of wheel movement for 19 movie segments (10,000 frames each). The shaded purple indicate an example of occluded wheel (see also Supplemental Data A.6). (f) Same as (a–d) for whisker inferred movement and ground truth. (g) Correlation coefficient of estimated vs ground truth magnitude of whisker movement in the case of semiautomatic (top) and automatic (bottom) inference for 6 movie segments (740 frames each). The shaded circles indicate examples containing frames with occlusion (see also Supplemental Data A.6). (h) Analogously to (a–f) spatial (left) and temporal (right) components corresponding to hindlimb (top) and forelimb (bottom) extracted by Flow-Loc from videos capturing a lateral mouse view. Notice in the bottom temporal component the alternating pattern of the gait. (i) As in panel (g) correlation coefficient between ground truth and estimated magnitude for the semiautomatic (top) and automatic (bottom) case. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

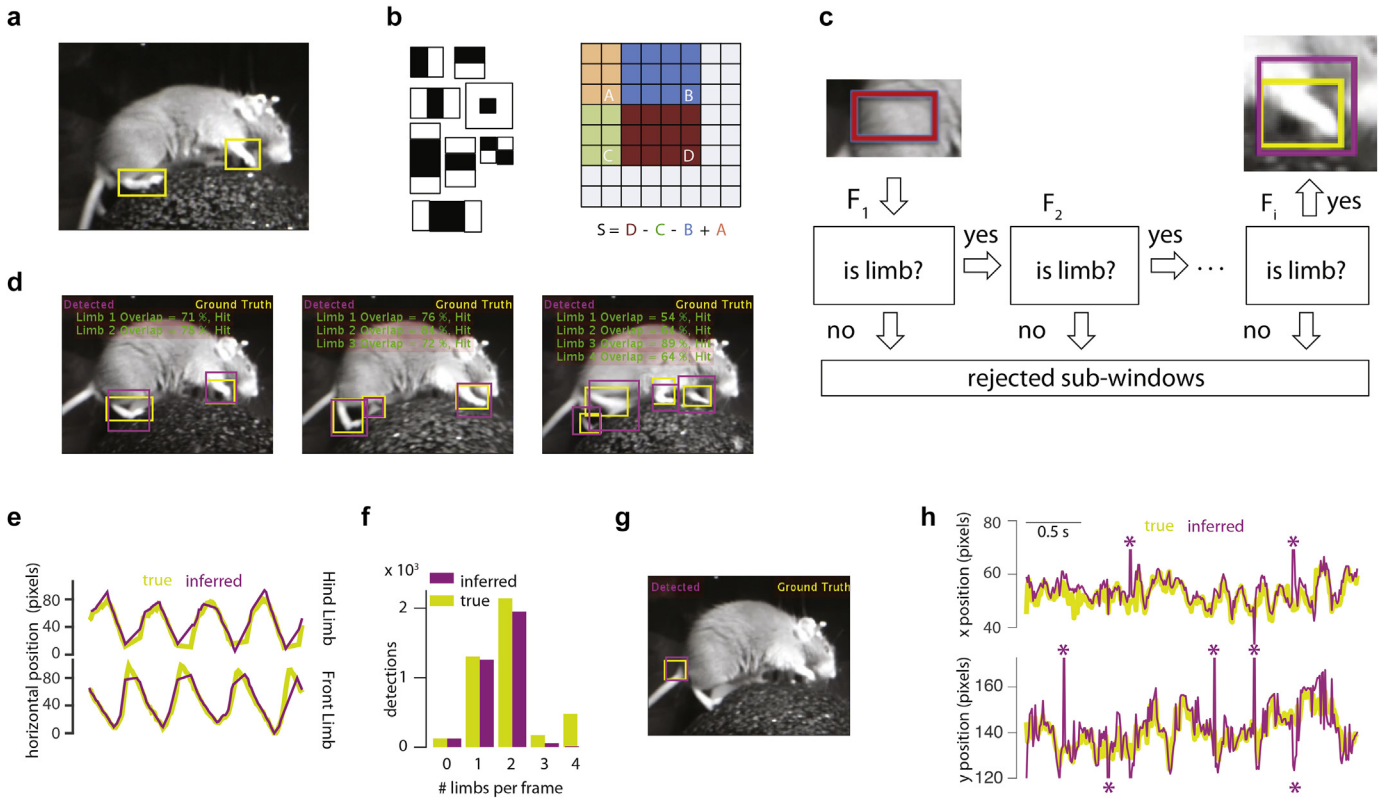

Fig. 4. Block-Loc. (a) Illustration of the mouse with two examples of manually labelled paws (yellow squares). (b) Left: Convolution of a bank of rectangular filters (Haar filters, left). This step is optimized using the integral image, which takes benefit of the rectangular structure of the Haar filters. The integral image (right) is constructed in such a way that each pixel (i, j) in the integral image is the result of adding all the image pixels from the previous pixels in both coordinates. The convolution with a rectangular filter becomes a simple addition of four numbers. (c) Cascade of boosted classifiers. All the cropped windows are extracted from the image following a sliding-windows approach, and the Haar features are computed. Then the sub-windows are processed in a cascade of classifiers that discard the vast majority of crops at early stages. (d–f) Empirical evaluation of Block-Loc. (d) Examples of overlaid detected and manually identified limbs in the case of 2 (left), 3 (middle) and 4 (right) detected limbs. The degree of overlap is describe in Section 3.3.1. (e) *Left*: Overlaid ground truth (yellow) and inferred (purple) horizontal position for hindlimb (top) and forelimb (bottom). (f) Histogram describing the number of limbs detected per frame. In purple the detected and in yellow the ground truth. (f) Examples of overlaid detected and manually identified tail location. (h) Overlaid ground truth (yellow) and inferred (purple) horizontal (top) and vertical (bottom) positions for tail. Purple asterisk indicates time points where the detected location was substantially far from the ground truth. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

can be expressed in terms of sums of rectangular areas of pixels, and can be reduced to three fixed-point operations (two subtractions and one addition) on the integral image regardless of the size of the filter. The resulting features are used to train a cascade of classifier ensembles. The Adaboost (Freund and Schapire, 1995) algorithm is used to train a robust combination of weak decision stump classifiers on the Haar features within a sliding window (Fig. 4(c)). Each classifier is trained to discard a large number of sub-windows at early stages of the cascade, and the final classifiers are specialized to process the most difficult examples. At the detection phase, only a few sub-windows pass through all layers of the cascade, and each sub-window is classified as limb/no limb or tail/no tail.

From a training set with 8013 manually annotated limb and 1820 tail locations, we used 1000 (paw) and 1052 (tail) samples to train the cascade of classifiers, and reserved the remaining samples for testing. The detection results are expressed in terms of the algorithm's precision and recall. Predicted bounding boxes (B_p) are considered as true positives (TP) if the overlap ratio (or Jaccard index) between the areas of B_p and the ground truth (B_g) exceeds 0.5:

$$ratio = \frac{|B_p \cap B_g|}{|B_p \cup B_g|}. \quad (4)$$

The precision and recall are defined as:

$$\text{precision} = \frac{TP}{TP + FP}, \quad \text{recall} = \frac{TP}{TP + FN}$$

Both measures can be combined into a single metric using the F_1 score:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (5)$$

2.5. Generation of ground truth data

2.5.1. Whisker movement

We generated ground truth data to evaluate whisker movements by labeling 6 movie sections (720×6 frames) from 3 mice. For each frame, we manually marked a fixed point on a selected whisker, then carefully tracked the trajectory of this point. Three of the six movies included frames where there was partial or total whisker occlusion due to grooming. The average labeling time for whiskers was 1.2 ± 0.5 s per frame.

2.5.2. Hindlimb movement

We generated ground truth data to evaluate hindlimb movements by labeling 2 movie sections (720×2 frames) with a lateral view of one mouse. For each frame we manually marked a fixed point on the tip of the mouse hindlimb and then carefully tracked

the movement of this point. The average labeling time for hind-limb was 0.8 ± 0.5 s per frame.

2.5.3. Paw and tail location

We extracted ground truth samples for tail and paw locations by manually annotating the tail and all visible limbs in 7 movie sections (Fig. 4(a) and (g)). At each frame we precisely annotated the top left corner and the width and height of the window containing the paw or tail (window centered at the superior pixel that joins the tail with the body, since the tip of tail is not visible). The database contains 4217 frames and a total of 8013 limb and 3337 tail windows were annotated. For paw annotation, the size and number of windows containing limbs differ depending on the phase within the stride cycle. The labeling took 10 h and 4 h for paws and tail respectively (amounting to about 8 and 4 s per frame).

2.5.4. Wheel movement

In what follows we present an extended and more detailed explanation of the method we previously employed in Giovannucci et al. (2017) to measure locomotion in head-fixed mice, and here taken as ground truth. In order to generate estimates of wheel movement from frontal camera perspective (Fig. 1(b) right), we employed a model-based semi-automated image processing algorithm. This method relies on measured distances between physical features on the wheel in order to adjust for camera perspective and eventually to convert image coordinates into physical units of displacement.

The primary salient feature on the wheel is a regularly repeating grid of circular "nubs" on its surface. We measured their radii to be 1.875 mm with 5.625 mm center-to-center spacing between nubs within the same column and 2.8125 mm vertical offset between adjacent columns (Extended Data A.5a). We generated sets of points spaced proportionally to match the contours of the wheel nubs. Using an interactive interface, these points were then overlaid onto a sample video from each experiment that was recorded from a different perspective, and we manually translated the plane of contours to match the position of the circles on the wheel in the video (Extended Data A.5b). This calibration procedure allowed us to compute a projective transform T that maps points on the camera's imaging plane X_{cam} to those in the isotropically spaced grid X_{iso}:

$$X_{cam} \times T = X_{iso} \tag{6}$$

where X_{cam} and X_{iso} are sets of point correspondences of the form $\begin{bmatrix} x & y & 1 \end{bmatrix}$ and T is a 3×3 matrix of the form:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

The projective transform, also known as a *homography*, is a mapping from one plane to another that preserves collinearity between points, but unlike an affine transform, it does not constrain parallel lines to remain parallel after the transformation. By applying this transform to each frame of the movie, we rectified the perspective of the camera such that distances in pixel space are proportional to the measured model parameters, i.e., after projection, the wheel nubs that were further from the camera were the same size as nubs that were closer. Although the wheel is not planar, this rectification allows for an approximation that is accurate enough for local displacement estimation.

After correcting for the camera perspective, the images were projected onto a plane such that the wheel displacement was proportional to the vertical displacement of the nubs in pixel space (Extended Data A.5). To track these circles, we first preprocessed the images by applying Contrast Limited Adaptive Histogram Equalization (CLAHE) to each frame to account for inhomogeneities in illumination, and to enhance edge contrast (Zuiderveld, 1994). Then, we used the Histogram of Oriented Gradients (HOG) descriptor (Dalal and Triggs, 2005) to compute a representation of the appearance of regions across the image (Extended Data A.5c). These descriptors represent the appearance of an image patch by concatenating histograms of the intensity gradient across multiple angular directions in subregions of the patch into a fixed length vector. This vectorized description can then be used to measure the similarity between image patches with robustness to small appearance changes.

Once HOG features were extracted from a pair of subsequent frames, matching points in the image were identified by computing the Euclidean distance between their corresponding HOG descriptors and performing a nearest neighbor search (Extended Data A.5d, top). As multiple parts of the image may look similar and lead to spurious matches, we computed the ratio between the distance of a point and its two nearest neighbors and removed those with a ratio greater than 0.7. To increase robustness to noise, e.g., due to feature localization, we compute the distribution of displacements and use the median vertical displacement as the true estimate for displacement (Extended Data A.5d, bottom). Frames with too few matches (<5) were excluded and displacement was linearly interpolated between frames with sufficient matches for a robust estimate.

Finally, as the projected pixel space is isotropic and proportional to physical space, the median pixel displacement can also be scaled by a constant factor based on the model parameters to yield the final estimate of instantaneous physical displacement of the wheel (Extended Data A.5e).

2.6. Computing details

All the Flow-Loc computations were performed on a Fedora linux Dell Precision Tower 7910, 24 cores Intel(R) Xeon(R) E5-2643 v3 @3.40 GHz, 128 GB RAM. Computation was restricted to a single core for evaluation purposes. All the Block-Loc training and testing was performed on a dedicated desktop machine (intel® i7 3.30 GHz, 16 GB RAM). All the code for Flow-Loc and Block-Loc was written in Python.

2.7. Data and code sharing

The labeled data employed to train the algorithms along with the corresponding code will be made publicly available.

3. Results

In this section we discuss the advantages of the software developed for recording videos of behaving mice. Then we apply the two algorithms for quantifying mouse behavior to example videos and present their performance in comparison to manually labeled data.

3.1. High-speed video acquisition of mouse behavior

We developed a Python package (https://github.com/bensondaled/eyeblink) for the acquisition of high-speed videos of mouse behavior from multiple cameras (Fig. 1). The setup is inexpensive since it only requires Playstation Eye cameras (8 $ each), a computer, and a data acquisition board (<$500, optional). Given enough computing power, the system can operate at speeds up to 187 Hz. In our experiments (Fig. 1(a)) we used one camera at a time acquiring at 100 Hz. The software (see Section 2.2) also allows the user to set various stimulation parameters, visualize the recorded frames, and plot some basic behavioral features like overall mouse movement as estimated from the pixel variability

(Fig. 1(c)). Moreover, the setup is amenable to closed loop applications, such as delivery of stimuli according to the behavioral status of the animal (e.g., deliver stimulus only if the mouse is not running).

3.2. Flow-Loc: flexible and fully automatic extraction of mouse movements

We developed an algorithm that focuses on measuring movements that repeatedly span a localized region in the FOV (see Section 2.3 for more details). The software is packaged as part of an open source toolbox (https://github.com/simonsfoundation/CaImAn) and is available for use to the scientific community. The algorithm provides a turn-key solution for practicing neuroscientists since it can scale to large datasets and only takes one parameter, the number of components. This parameter roughly corresponds to the number of detectable movement patterns present in the video (see Section 2.3.1 for more details). Intuitively, the algorithm tries to identify and cluster groups of pixels within the FOV that move in similar directions. The input of the algorithm (number of components) then roughly corresponds to the number of such clusters, and the output represents both the spatial extent and the time course of the velocities associated with these clusters (see Fig. 2(c) and (d)). For easier interpretation, the time course of the velocities are expressed in polar coordinates, providing velocity magnitudes (black traces in Fig. 2) and angles (red traces). Given the way we model the clusters (see Section 2.3, matrix factorization through dictionary learning), the algorithm can only capture movements that repeatedly and consistently span the same area within the FOV. The intuition is that we associate fixed groups of pixels to movements, and therefore movements that roam across different portions of the FOV are poorly captured. In Fig. 2(c) and (d) we depict the results of Flow-Loc applied to videos captured from a side view of the mouse; the spatial components capture the location of specific mouse/object portions (limbs, trunk, wheel) whereas the temporal components capture the oscillatory nature of gait (shaded gray lines). Below we provide a thorough empirical evaluation of the accuracy and computational efficiency of Flow-Loc.

3.2.1. Empirical evaluation of accuracy in detecting locomotion speed, whiskers, and limb movements

We empirically tested Flow-Loc on movies collected from two perspectives, the front and side views of the mouse on a treadmill (see Fig. 2), and under two user interaction levels, semiautomatic (user selects the specific area of the animal to be monitored) or completely automatic. The movements we considered in our evaluation are: general locomotion inferred by wheel movement, whisking, and gait (hindlimb). In order to benchmark the Flow-Loc algorithm we extracted ground truth data from frames either manually (whiskers and hindlimb position, Section 2.5.1) or by relying on clear and easily identifiable landmarks (wheel velocity, Section 2.5.4). We then compared the time course similarity of the ground truth velocities with the velocities estimated by Flow-Loc (Fig. 3). The metric we employed to quantify the similarity of inferred and ground truth movement is the Pearson's correlation coefficient (r) between the estimated and ground truth time courses of the velocity magnitudes. We restricted the comparison to the magnitude only (discarding the direction) since the angle becomes very noisy for small values, as mentioned in Section 2.3. Given that the only input to the algorithm is the number of components (see Section 2.3.1 concerning the selection of this number), we varied this parameter and measured its effect on algorithm performance. For experiments in the completely automatic setting, we used up to five components for the frontal view and up to six components for the side view. For the semi-automatic case, we used up to 3 components, except for the wheel, where one component was sufficient (see Section 2.3.1 for the rationale of this choice). When the number of components was larger than one, we report the maximum among all r values.

Table 1
Evaluation of Flow-Loc against ground truth. The reported values represent the average correlation of the components best matching the ground truth.

Num components	Wheel	Whisker	Hindlimb
(a) Semi-automated case			
1	**0.85 ± 0.17**	0.72 ± 0.18	0.57 ± 0.03
2	–	0.75 ± 0.18	0.81 ± 0.03
3	–	0.74 ± 0.19	0.8 ± 0.01
(b) Automated case			
1	0.62 ± 0.31	0.65 ± 0.13	–
3	0.76 ± 0.18	0.74 ± 0.16	–
4	–	–	0.68 ± 0.01
5	0.82 ± 0.19	**0.78 ± 0.15**	0.76 ± 0.06
6	–	–	**0.85 ± 0.01**

Table 2
Computational performance on 40k frames for Flow-Loc (time in s).

Frame size	Load	Optical flow	DL	Per frame
100 × 200	87	257	286	0.015
320 × 240	87	833	373	0.032

In Table 1a and b and Fig. 3 we report the results for all our comparisons. First, note the agreement between the inferred speed and ground truth for a set of example components (Fig. 3(b), (d), and (h)), for both velocity magnitude alone (black and orange trace) and for velocity direction (cyan and red traces). In Fig. 3(e), (g), and (i) we run the comparison for multiple movies and animals. The r values are reported for two cases: a semi-automated case where the user selected a small ROI around a specific body part, and a fully automated case that operates without any human intervention (ROI including multiple body parts, or the full field of view). The results indicate that the average performance is similar in the semi- and fully automated cases, with the fully automated case resulting in better outcomes for whisker and hindlimb movements (Table 1a versus b).

Second, we observe that occlusion plays an important role in determining the accuracy of the algorithm (shaded circles in Fig. 3(e) and (g) indicate occlusion). Although the algorithm's performance is worse when occlusion is present, adding more components improves performance (Table 1a and b). In Fig. A.6 and movie M.1 we show in detail the effect of occlusion on the output of Flow-Loc. When the animal grooms the face with the paw, it is difficult for the algorithm to keep track of the underlying whiskers.

3.2.2. Empirical evaluation of computational performance

One of the most important features of Flow-Loc is its scalability. By operating in an online mode (see Section 2 for details), Flow-Loc does not process all the data at once, but it can incrementally provide and update its motion pattern estimates after analyzing chunks of the dataset. As a result, the algorithm does not need to load the full movie into memory, but can instead proceed by loading small chunks at a time. Therefore, even very large datasets can be processed on medium-sized machines. Moreover, the algorithm is highly optimized and converges rapidly (Mairal et al., 2010). In Table 2 we report the computing time required to run Flow-Loc on 40000 frames, partitioned by time required to load the movie, time to calculate the optical flow, and time to solve the dictionary learning problem. Notice that the overall speed for processing the dataset amounts to roughly 30 and 60 Hz for the two cases in Table 2, slower than real time (100 Hz). Additionally, as noted in Section 2.3, Flow-Loc cannot operate on one frame at a time. Therefore, closed loop experiments are currently unfeasible under this framework.

Table 3
Precision, recall, accuracy and F_1 measure of the detection experiments.

Method	Precision	Recall	Accuracy	F_1
Block-Loc (paw)	100	81	91.2	89.6
Block-Loc (tail)	99.5	86.7	92.7	86.4

3.3. Block-Loc: fast cascade of classifiers for automated object detection

Flow-Loc provides a handy turn-key unsupervised system for the analysis of behavior. However, it is limited in two ways: it cannot be applied to real time closed loop experiments, and it cannot track objects that span large areas in the FOV without a consistent spatial pattern (i.e. for instance paw while grooming). In this section we present a fast algorithm (Block-Loc) that relies on efficient filtering to process video data faster than real time. Paw/tail localization and tracking are performed through a fast robust object detection approach, similar to Viola and Jones (2001, 2004), that is not limited to spatially stereotyped patterns but nevertheless requires supervised training with labeled data. Intuitively, the algorithm divides the FOV into overlapping rectangular windows of varying sizes, then trains a classifier to decide whether any of these rectangles contain the target object (paw/limb) (see Section 2.4 for details about the algorithm). In the following section we detail our empirical evaluation of the accuracy and computational performance of Block-Loc.

3.3.1. Empirical evaluation of accuracy on paw and tail detection

We ran the evaluation on movies capturing a lateral view of the mouse. The goal was to train the algorithm to localize either paws or base of the tail (the point where the tail connects to the body, the only portion that is visible in our lateral movies). Roughly 1000 manually labeled rectangles were enough to train Block-Loc to achieve the performance reported below. In movie M.2 (paw) and M.3 (tail) we present examples of the trained classifiers running on test videos. Note in M.2 how the algorithm identifies the ipsilateral paws but exhibits some inaccuracy with the less-visible contralateral paws. Table 3 displays the results of the benchmark for both cases. In the case of paw detection, the algorithm does not detect any false positive, and it misses less than 20% of the detectable paws, thus reaching an accuracy of more than 90% (see Section 2.4 for details about the metrics). In the case of tail detection, the algorithm displays similar performance.

Fig. 4(d)–(h) presents some example results on paws and tail. In some cases, the algorithm is able to detect the paws on both sides of the mouse (Fig. 3(d), right). However, this is not always the case (Fig. 4(f)) as limbs in the foreground tend to significantly occlude the background paws, causing false negatives. On the other hand, paws closer to the camera are properly detected in most instances. For example, Fig. 3(e) illustrates that detection of the paws in the foreground is consistent across frames capturing 4 strides, even though no constraint of such type is imposed. For what concerns the other mouse portion tested, Fig. 3(g) presents an example frame of Block-Loc applied to detection of the tail. In the case of tail detection across time, in some frames the detection is off by few pixels (asterisks in Fig. 3(h)). This can be improved by considering spatial consistency across time (see Section 4).

3.3.2. Evaluation of computational performance

We evaluated the computational performance of Block-Loc by measuring the time required to acquire the labeled data, train the algorithm, and apply it to single frames. In order to train the algorithm, users need to manually mark rectangles around target objects (paws, limbs, etc.). The training examples required to train the paw and tail classifiers were collected in two single sessions of 10 h and 4 h, respectively. The algorithm was trained on a medium-sized machine for 2 h 57 min (paw) and less than 1 h for the tail (see Section 3.3.1). Since the algorithm leverages an efficient method for creating and evaluating sliding windows, it can operate at speeds faster than real time (5.66 ms per frame, roughly 177 Hz). Therefore, videos recorded at 100 frames per second were processed faster than real time (0.56 s to process 100 frames). Therefore, the computational efficiency of Block-Loc deems it suitable for closed loop applications.

4. Discussion

The growing field of computational ethology (Anderson and Perona, 2014) seeks to automate the measurement and analysis of animal behavior. Identifying motor primitives exclusively by human observation and manual labeling is often impractical, and also suffers from subjectivity and irreproducibility. The automatic analysis of behavior can be approached at varying levels of description, ranging from the identification and clustering of stereotypical behavioral modules (Wiltschko et al., 2015; Berman et al., 2014; Vogelstein et al., 2014; Kabra et al., 2013) to the quantitative characterization of movement kinematics (Clack et al., 2012; Machado et al., 2015; Heiney et al., 2014a; Lee et al., 2015; Nashaat et al., 2017).

In this work we focused on estimating the kinematics of movements in the specific case of the head-fixed mouse preparation, in which the animal's head is anchored to a fixed point. This preparation, because of its stability and versatility, is currently a standard for a number of recording and stimulation techniques in neuroscience. With this goal in mind, we developed an open-source package that enables high speed video recordings from multiple cameras by relying on inexpensive hardware. Furthermore, we presented two algorithms to analyze the acquired movies. The first, Flow-Loc, is a flexible and automated approach to the estimation of spatially stereotyped motion patterns in head-fixed mice. The second, Block-Loc, is a general-purpose supervised algorithm for detecting features of the mouse body, that can operate in faster than real time and can be used to facilitate closed loop experiments. We evaluated both approaches by comparing their performance against ground truth data and demonstrated good performance for both (correlations with ground truth speed in the range 0.75–0.85 for Flow-Loc and accuracy levels above 90% for Block-Loc). Flow-Loc and Block-Loc employ reliable, efficient, and advanced computational tools. Flow-Loc builds upon dense optical flow (Farnebäck, 2003), an algorithm used to estimate velocity fields, and online dictionary learning (Mairal et al., 2010), a dimensionality reduction method. Their interplay introduces several advantages. First, by estimating the optical flow, it operates in the velocity manifold, convenient for analyzing motion through DL, which automatically identifies spatial footprints and temporal dynamics of the motion patterns, preventing to some degree the interference of occlusions. Both DL and optical flow are efficiently implemented in optimized open source libraries (Mairal et al., 2010; Itseez, 2015), thus delivering excellent computational performance. Block-Loc achieves fast training time and faster than real time speeds in detecting objects using cascades of fast classifiers on Haar features (position inference in 5 ms, initial training in less than 3 h and performed only once). At the detection step, the algorithm can operate in real time even in high frame rate settings.

4.1. Related work

Many automated solutions for measuring animal behavior are emerging in the field of computational ethology for both invertebrate and vertebrate preparations. A large number of these

approaches focus on freely-moving animals or on detecting high-level behaviors (Wiltschko et al., 2015; Berman et al., 2014; Vogelstein et al., 2014; Kabra et al., 2013; Machado et al., 2015) rather than specific movement kinematics in head-fixed mice. Even though some open-source and turn-key frameworks exist for the freely-behaving preparation (Patel et al., 2014; Ben-Shaul, 2017; Dagan et al., 2016), to the best of our knowledge no general purpose set of tools for measuring behavior exists for the head-fixed preparation. For this case, many custom solutions have been proposed in the literature. Some approaches rely on markers (Zörner et al., 2010; Lee et al., 2015) or tattooed color landmarks (Nashaat et al., 2017) to facilitate the identification of mouse movements. Other approaches have relied on specialized solutions based on customized experimental setups or equipment, like the fully automated tracking of vibrissae presented in Clack et al. (2012), which requires expensive hardware and specific illumination and position conditions, as well as the presence of a single row of whiskers. Other examples include the work of Hoogl et al. (2015), where a transparent spinning disk is required to monitor mouse behavior from underneath, or the supervised learning approach used in Guo et al. (2015) where paws are tracked by cascaded pose regression from two simultaneous camera recordings that are used to quantify prehension. Finally, unsupervised learning algorithms have also been used, based on the estimation of pixel intensity variability within a manually selected region of interest (Schneider et al., 2014) or on principal component analysis (PCA) of a custom linear transformation of the pixel intensity space (Powell et al., 2015).

Our contribution is a turn-key, general framework that can be directly employed to acquire and quantify mouse movement kinematics in head-fixed settings, requiring either just one tunable parameter (Flow-Loc), or a single set of training samples to reach reasonable accuracy in movement quantification (Block-Loc). This performance is achieved with scalable and fast algorithms.

4.2. Movements that Flow-Loc and Block-Loc can track

Flow-Loc, given its underlying mathematical implementation, is not applicable in its current form to measuring behavior in freely-behaving preparations. Additionally, since both Flow-Loc and Block-Loc focus on measuring kinetics, they cannot deal with quantification and classification of abstract behaviors composed of several motor syllables, such as grooming. However, these algorithms will likely generalize to other movements captured at proper speed and magnification, such as licking or eye movements.

4.3. Granularity of movements captured by Flow-Loc

One question emerging from our work concerns the spatial granularity of the motion information captured using Flow-Loc. For instance, when measuring hindlimb velocity, see Fig. 3, what portion of the limb (paw, leg, thigh) is associated with the velocity measure? The algorithm measures the average velocity field of a set of pixels, so the most accurate interpretation would be an average of the entire limb. In the case of whiskers, the detected temporal components do not correspond to single whiskers, but rather to an estimate of the average movement of the whisker population spanned by the corresponding spatial component.

4.4. Future work

In the future, we plan to extend the Flow-Loc algorithm presented here to run in an online streaming mode (processing one frame at a time), since the optical flow computation is inherently amenable to frame-by-frame analysis. Refinements may also be attempted for the Block-Loc algorithm. For instance, in the detection step, similarly to Guo et al. (2015), one could use time coherence constraints to smooth the object detection trajectories in time. Similarly, we could significantly speed up detection by constraining the search to the neighborhood of features detected in previous frames. Furthermore, the same temporal coherence could be applied to interpolate the missed paw detections from consecutive frames (increasing the recall of the method).

4.5. Impact on the neuroscience community

The head-fixed mouse preparation is receiving considerable attention within the neuroscience community, thanks to its desirable properties associated with stable recordings, stimulation, and virtual reality setups. The presented set of tools is a useful complement to such experimental preparations, providing a low-cost, flexible, and efficient turn-key solution to the challenge of quantitatively measuring behavior, both in offline analysis and in closed loop experimental settings.

Authors' contributions

A.G. designed behavioral + imaging experiments, established the behavioral and imaging set-up, performed experiments, developed analyses. A.G. and D.M wrote the manuscript. A.G. and E.P. developed the Flow-Loc algorithm. D.M., W.A., P.P. developed the Block-Loc detection algorithm. B.D. developed the software for camera acquisition. T.P. developed the algorithm for wheel ground truth generation. J.F. and M.B. helped with experiments. S.W. designed experiments and developed analyses. All authors edited the manuscript. The authors thank Laura Lynch for expert laboratory assistance.

Acknowledgements

This work was supported by National Institutes of Health grant R01 1NS045193 (S.W.), New Jersey Commission on Brain Injury Research fellowship CBIR12FEL031 (A.G.), Nancy Lurie Marks Family Foundation (S.W.), TIN2015-66951-C2-2-R grant from the Spanish Ministry of Economy and Competitiveness and FEDER, UOC doctoral research grant and NVidia Hardware grant programs (D.M.), National Science Foundation Graduate Research Fellowship4DGE-1148900 (T.P.).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.jneumeth.2017.07.014.

References

Anderson, D.J., Perona, P., 2014. Toward a science of computational ethology. Neuron 84, 18–31.
Ben-Shaul, Y., 2017. Optimouse: a comprehensive open source program for reliable detection and analysis of mouse body and nose positions. BMC Biol. 15, 41.
Bengio, S., Pereira, F., Singer, Y., Strelow, D., 2009. Group sparse coding. In: Advances in Neural Information Processing Systems., pp. 82–89.
Berman, G.J., Choi, D.M., Bialek, W., Shaevitz, J.W., 2014. Mapping the stereotyped behaviour of freely moving fruit flies. J. R. Soc. Interface 11, 20140672.
Chettih, S.N., McDougle, S.D., Ruffolo, L.I., Medina, J.F., 2011. Adaptive timing of motor output in the mouse: the role of movement oscillations in eyelid conditioning. Front. Integr. Neurosci. 5, 72.
Clack, N.G., O'Connor, D.H., Huber, D., Petreanu, L., Hires, A., Peron, S., Svoboda, K., Myers, E.W., 2012. Automated tracking of whiskers in videos of head fixed rodents. PLoS Comput. Biol. 8, e1002591.
Dagan, S.Y., Tsoory, M.M., Fainzilber, M., Panayotis, N., 2016. Colorcation: a new application to phenotype exploratory behavior models of anxiety in mice. J. Neurosci. Methods 270, 9–16.
Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, CVPR 2005., pp. 886–893, IEEE volume 1.

Dombeck, D.A., Graziano, M.S., Tank, D.W., 2009. Functional clustering of neurons in motor cortex determined by cellular resolution imaging in awake behaving mice. J. Neurosci. 29, 13751–13760.
Dombeck, D.A., Khabbaz, A.N., Collman, F., Adelman, T.L., Tank, D.W., 2007. Imaging large-scale neural activity with cellular resolution in awake, mobile mice. Neuron 56, 43–57.
Farnebäck, G., 2003. Two-frame motion estimation based on polynomial expansion. In: Scandinavian Conference on Image Analysis. Springer, pp. 363–370.
Freund, Y., Schapire, R.E., 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory. Springer, pp. 23–37.
Giovannucci, A., Badura, A., Deverett, B., Najafi, F., Pereira, T., Gao, Z., Ozden, I., Kloth, A., Pnevmatikakis, E., Paninski, L., et al., 2017. Cerebellar granule cells acquire a widespread predictive feedback signal during motor learning. Nat. Neurosci. 20, 727–734.
Guo, J.-Z., Graves, A.R., Guo, W.W., Zheng, J., Lee, A., Rodríguez-González, J., Li, N., Macklin, J.J., Phillips, J.W., Mensh, B.D., et al., 2015. Cortex commands the performance of skilled movement. Elife 4, e10774.
Harvey, C.D., Collman, F., Dombeck, D.A., Tank, D.W., 2009. Intracellular dynamics of hippocampal place cells during virtual navigation. Nature 461, 941–946.
Heiney, S.A., Kim, J., Augustine, G.J., Medina, J.F., 2014a. Precise control of movement kinematics by optogenetic inhibition of Purkinje cell activity. J. Neurosci. 34, 2321–2330.
Heiney, S.A., Wohl, M.P., Chettih, S.N., Ruffolo, L.I., Medina, J.F., 2014b. Cerebellar-dependent expression of motor learning during eyeblink conditioning in head-fixed mice. J. Neurosci. 34, 14845–14853.
Hira, R., Honkura, N., Noguchi, J., Maruyama, Y., Augustine, G.J., Kasai, H., Matsuzaki, M., 2009. Transcranial optogenetic stimulation for functional mapping of the motor cortex. J. Neurosci. Methods 179, 258–263.
Hira, R., Terada, S.-I., Kondo, M., Matsuzaki, M., 2015. Distinct functional modules for discrete and rhythmic forelimb movements in the mouse motor cortex. J. Neurosci. 35, 13311–13322.
Hoogland, T.M., De Gruijl, J.R., Witter, L., Canto, C.B., De Zeeuw, C.I., 2015. Role of synchronous activation of cerebellar Purkinje cell ensembles in multi-joint movement control. Curr. Biol. 25, 1157–1165.
Itseez, 2015. Open Source Computer Vision Library, https://github.com/itseez/opencv.
Kabra, M., Robie, A.A., Rivera-Alba, M., Branson, S., Branson, K., 2013. Jaaba: interactive machine learning for automatic annotation of animal behavior. Nat. Methods 10, 64–67.
Kloth, A.D., Badura, A., Li, A., Cherskov, A., Connolly, S.G., Giovannucci, A., Bangash, M.A., Grasselli, G., Pe nagarikano, O., Piochon, C., et al., 2015. Cerebellar associative sensory learning defects in five mouse autism models. Elife 4, e06085.
Lee, K.H., Mathews, P.J., Reeves, A.M., Choe, K.Y., Jami, S.A., Serrano, R.E., Otis, T.S., 2015. Circuit mechanisms underlying motor memory formation in the cerebellum. Neuron 86, 529–540.
Lienhart, R., Maydt, J., 2002. An extended set of Haar-like features for rapid object detection. In: International Conference on Image Processing, 2002, Proceedings, vol. 1. IEEE, pp. I-900.
Machado, A.S., Darmohray, D.M., Fayad, J., Marques, H.G., Carey, M.R., 2015. A quantitative framework for whole-body coordination reveals specific deficits in freely walking ataxic mice. Elife 4, e07892.
Mairal, J., Bach, F., Ponce, J., Sapiro, G., 2010. Online learning for matrix factorization and sparse coding. J. Mach. Learn. Res. 11, 19–60.
Najafi, F., Giovannucci, A., Wang, S.S.-H., Medina, J.F., 2014. Sensory-driven enhancement of calcium signals in individual Purkinje cell dendrites of awake mice. Cell Rep. 6, 792–798.
Nashaat, M.A., Oraby, H., Pe na, L.B., Dominiak, S., Larkum, M.E., Sachdev, R.N., 2017. Pixying behavior: a versatile real-time and post hoc automated optical tracking method for freely moving and head fixed animals. eNeuro 4, ENEURO-0245.
Olshausen, B.A., Field, D.J., 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381, 607.
Owen, A.B., Perry, P.O., 2009. Bi-cross-validation of the SVD and the nonnegative matrix factorization. Ann. Appl. Stat., 564–594.
Patel, T.P., Gullotti, D.M., Hernandez, P., O'Brien, W.T., Capehart, B.P., Morrison III, B., Bass, C., Eberwine, J.E., Abel, T., Meaney, D.F., 2014. An open-source toolbox for automated phenotyping of mice in behavioral tasks. Front. Behav. Neurosci. 8.
Powell, K., Mathy, A., Duguid, I., Häusser, M., 2015. Synaptic representation of locomotion in single cerebellar granule cells. Elife 4, e07290.
Schneider, D.M., Nelson, A., Mooney, R., 2014. A synaptic and circuit basis for corollary discharge in the auditory cortex. Nature 513, 189–194.
Sofroniew, N.J., Cohen, J.D., Lee, A.K., Svoboda, K., 2014. Natural whisker-guided behavior by head-fixed mice in tactile virtual reality. J. Neurosci. 34, 9537–9550.
Viola, P., Jones, M., 2001. Robust real-time object detection. Int. J. Comput. Vis., 4.
Viola, P., Jones, M.J., 2004. Robust real-time face detection. Int. J. Comput. Vis. 57, 137–154.
Vogelstein, J.T., Park, Y., Ohyama, T., Kerr, R.A., Truman, J.W., Priebe, C.E., Zlatic, M., 2014. Discovery of brainwide neural-behavioral maps via multiscale unsupervised structure learning. Science 344, 386–392.
Wiltschko, A.B., Johnson, M.J., Iurilli, G., Peterson, R.E., Katon, J.M., Pashkovski, S.L., Abraira, V.E., Adams, R.P., Datta, S.R., 2015. Mapping sub-second structure in mouse behavior. Neuron 88, 1121–1135.
Witter, L., Canto, C., Hoogland, T.M., De Gruijl, J.R., De Zeeuw, C.I., 2013. Strength and timing of motor responses mediated by rebound firing in the cerebellar nuclei after Purkinje cell activation. Front. Neural Circuits 7, 133.
Zörner, B., Filli, L., Starkey, M.L., Gonzenbach, R., Kasper, H., Röthlisberger, M., Bolliger, M., Schwab, M.E., 2010. Profiling locomotor recovery: comprehensive quantification of impairments after CNS damage in rodents. Nat. Methods 7, 701–708.
Zuiderveld, K., 1994. Contrast limited adaptive histogram equalization. In: Graphics Gems IV. Academic Press Professional, Inc, pp. 474–485.