Received January 27, 2020, accepted February 10, 2020, date of current version March 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2975926

Limbs Detection and Tracking of Head-Fixed Mice for Behavioral Phenotyping Using Motion Tubes and Deep Learning

WASEEM ABBAS[1], DAVID MASIP[1], (Senior Member, IEEE), AND ANDREA GIOVANNUCCI[2,3]

[1]Network and Information Technologies, Universitat Oberta de Catalunya, 08018 Barcelona, Spain
[2]Joint Department of Biomedical Engineering, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA
[3]Joint Department of Biomedical Engineering, North Carolina State University, Raleigh, NC 27695, USA

Corresponding author: Waseem Abbas (abbas@uoc.edu)

This work was supported in part by the Spanish Ministry of Science, Innovation and Universities (FEDER funds) under Grant TIN2015-66951-C2-2-R and Grant RTI2018-095232-B-C22, and in part by the NVIDIA Hardware Grant Program.

ABSTRACT The broad accessibility of affordable and reliable recording equipment and its relative ease of use has enabled neuroscientists to record large amounts of neurophysiological and behavioral data. Given that most of this raw data is unlabeled, great effort is required to adapt it for behavioral phenotyping or signal extraction, for behavioral and neurophysiological data, respectively. Traditional methods for labeling datasets rely on human annotators which is a resource and time intensive process, which often produce data that that is prone to reproducibility errors. Here, we propose a deep learning-based image segmentation framework to automatically extract and label limb movements from movies capturing frontal and lateral views of head-fixed mice. The method decomposes the image into elemental regions (superpixels) with similar appearance and concordant dynamics and stacks them following their partial temporal trajectory. These 3D descriptors (referred as motion cues) are used to train a deep convolutional neural network (CNN). We use the features extracted at the last fully connected layer of the network for training a Long Short Term Memory (LSTM) network that introduces spatio-temporal coherence to the limb segmentation.
We tested the pipeline in two video acquisition settings. In the first, the camera is installed on the right side of the mouse (lateral setting). In the second, the camera is installed facing the mouse directly (frontal setting). We also investigated the effect of the noise present in the videos and the amount of training data needed, and we found that reducing the number of training samples does not result in a drop of more than 5% in detection accuracy even when as little as 10% of the available data is used for training.

INDEX TERMS Deep networks, motion detection, CNN, LSTM, optical flow, spatiotemporal, neuroscience, behavioral phenotyping.

I. INTRODUCTION

Neuroscience is becoming increasingly reliant on quantitative data. The availability of affordable computing and sensing methods has now made it possible to record milliseconds resolution videos of behaving mice while monitoring their neuronal activity simultaneously. A general trend in behavioral neuroscience is to record videos of the mice in a controlled but uninterrupted environment for extended periods of time. One of the commonly used controlled environments is to allow the mice walk on a spherical or cylindrical treadmill in a head-fixed position. This allows the mice to walk freely on the treadmill with minimal motion of their torso, while reducing the stress associated to head-fixation. The video cameras are usually placed either beside, above, below or in front of the mice. The acquired video data is annotated manually, frame by frame. As increasingly large volumes of data are generated, manual annotation becomes impractical for two reasons; the impossibility to scale up to hundreds of thousands of frames and the lack of reproducibility. Thus, high quality research calls for reliable automatic methods to replace manual labeling of animal behavior.

The associate editor coordinating the review of this manuscript and approving it for publication was Donato Impedovo.

Some of the commonly used methods for limb annotation either use specific hardware for motion tracking or model the statistical properties of the background and the animal (background subtraction). Hardware based approaches are usually difficult to reproduce in new scenarios. And background modelling is vulnerable to noisy images and moderate periods with lack of significant motion. Detecting only individual body parts of a moving animal becomes also challenging. In addition, due to motion and the change of perspective, limbs do not appear rigid throughout the video. They present relevant deformations from frame to frame. In these cases, traditional object recognition methods do not perform well. Robust appearance-based methods solve these shortcomings by learning a classifier [1] focused on diverse sets of image patches with the body part to be tracked. Here, we develop a pipeline that performs limb tracking by image segmentation, combining appearance-based features and temporal coherence. We use a spatio-temporal segmentation of mouse body parts relying on two neural networks. The first one, a convolutional neural network (CNN), learns features from sub-regions of the video which are similar in appearance and exhibit coherent motion patterns. We name these descriptors motion tubes. For example, a motion tube might encode the spatio-temporal representation of a mouse paw across a relatively short time window (See Figure1). We define motion sequence as the chronologically ordered concatenation of motion tubes, which in turn represents the time evolution of these coherent sequences. Finally, we use a Long-Short-Term-Memory (LSTM) network to predict the probability of these motion sequences of being part of the mice limbs. Experimental results show promising tracking accuracy even when only a small portion of training data is used. We also evaluate our approach against two state-of-the-art limb tracking algorithms, and show significant improvements, especially with noisy recordings.

II. RELATED WORK

Typical setups for motion tracking in neuroscience applications include a closed environment (either a room or a box), video cameras, an awake animal preparation and a set of control systems [2]. In this article, we will focus on motion tracking (with emphasis on limb tracking) of laboratory animals for behavioral phenotyping or medical assessment purposes. Based on the intended use and nature of algorithms, existing tracking approaches can be divided into three categories; hardware-based, software-based aided by specialized hardware and software-based without specialized hardware.

Kain *et al.* [3] proposed an explicit hardware-based leg tracking method for automated behavior classification in Drosophila flies. In such setup, the legs of head-fixed flies walking on a spherical treadmill are marked with fluorescent dyes. Multiple mounted cameras are then employed to track ad record 15 gait features in real-time. This approach has the advantage to enable real-time experiments, but it cannot generalize to other limb tracking applications unless a specific hardware setup is established. Moreover, the dependency on photo-sensitive dyes decrease the robustness of the system. Similar approaches are reported in [4]–[10].

Wang *et al.* [11] introduced a hardware-assisted pipeline for identifying micro-behaviors in small animals. This is achieved by employing Microsoft Kinect cameras along with regular video cameras to record movement of freely behaving rodents from three different perspectives. The IR depth images from the Kinect are used to extract the three dimensional shape of rodents by background subtraction. Five pixel-based features extracted from the resultant 3D blobs are fed into Support Vector Machines for behavior classification. Although the pipeline is not exclusively used for motion tracking, the use of depth cameras is a competitive alternative for motion tracking. This approach relies on specific hardware, therefore it cannot be applied in every environments since the Kinect motion sensors need specific ambient conditions for optimal performance.

Monteiro *et al.* [12] took a similar approach to [11] and used Microsoft Kinect depth cameras for video capture. Instead of background subtraction, they introduced a rough temporal context by encapsulating morphological features of multiple frames for motion tracking. Morphological features of each frame are then concatenated to introduce temporal context and used to train a decision tree for behavior classification. Similar to [11], this approach is also amenable for motion tracking, given the introduced temporal context.

Palmér *et al.* [13] introduced a paw-tracking algorithm for hand prehension and gesture tracking in mice. The algorithm is framed as a pose estimation problem. Each digit is modeled as a combination of three phalanges (bones), where each bone is fit to an ellipsoid. For 4 digits, there are a total of 12 ellipsoids. The palm and forearm are modeled by ellipses, whereas the forearm by an elliptic paraboloid. Therefore, 16 parameters define the paw digits (four degrees of freedom per digit), four constant vectors represent the metacarpal bones and 6 parameters describe the position and rotation of the palm of the paw. Furthermore, the forearm can rotate along all three axes around the wrist. This amounts to a total of 22 parameters. In each frame, these ellipsoids are fit to the body part edges. The article does not report any quantitative results. This approach is useful if the gesture tracking problem is treated as pose estimation with a temporal context. In subsequent work, Palmér *et al.* [14] reduced the degrees of freedom to 19 and optimized the computing strategy.

Mathis *et al.* [1] proposed a deep learning-based tool, called DeepLabCut, which estimated the pose of lab animals in each frame. Their proposed model is built on top of the DeeperCut algorithm [15]. The original DeeperCut approach was developed for multi-person pose estimation, which consists of three parts: object detectors, pairwise term and pose estimators. Object detectors are used for identifying body parts and pairwise terms exploit anatomical knowledge to develop a refined model of body segments. The pose estimator then translates this knowledge into pose. The paper proposes to add a pre-trained Deep Residual Network followed by deconvolutional layers on top of this architecture.

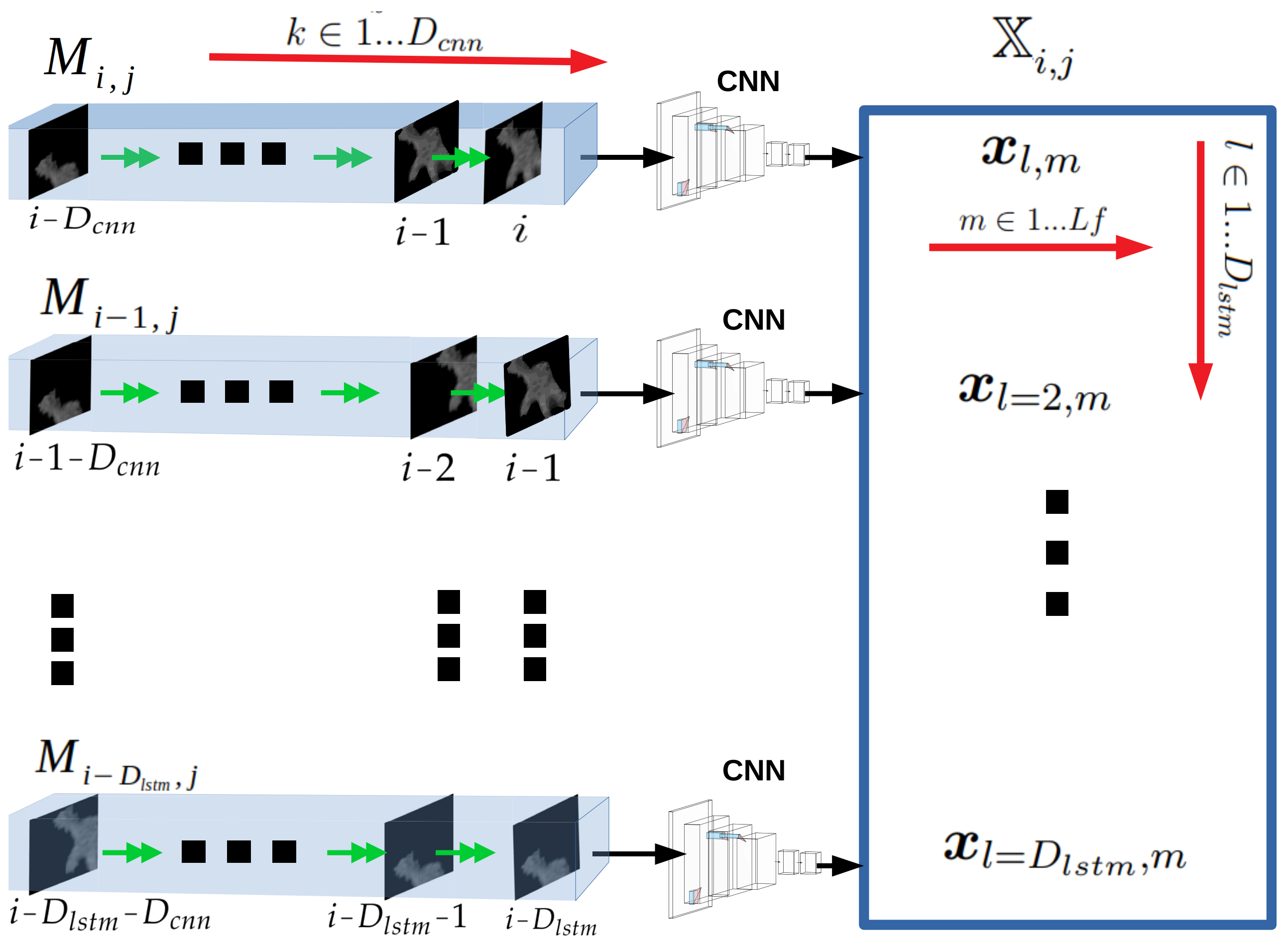


FIGURE 1. Master Flowchart: The image slice in the tubes $M_{i,j}$ represents a specific superpixel and the corresponding fixed window which encloses it in frame i and its various matches tracked through time. Double headed arrow represents the flow of time, so superpixel at time $i-1$ was the predecessor of superpixel at position i. The shaded 3D structures on the left represent the 3D motion tubes formed by stacking D_{cnn} superpixels together chronologically while the blue-bordered matrix on the right represents the motion sequence. An enlarged view of a motion tube with its constituent superpixel shaded in green is shown in Fig. 2. The individual feature vector extracted from the tube (formed by a specific superpixel number j from frame number i) is represented by a row vector of length L_f as $x_{l,m} = [x_{l,1}, x_{l,2}, \ldots, x_{l,L_f}]$. D_{cnn} represents the depth of motion tube and D_{lstm} represents the length of motion sequence which is given as input to the LSTM. A single motion sequence is formed by stacking feature vectors extracted from D_{lstm} motion tubes by the CNNs in correct chronological sequence.

The model is trained on manually annotated data of mice and drosophila flies. Consistent RMSE (error measured between location of body parts predicted by pipeline and human annotators) was achieved despite a 90% decrement in training frames. Due to its end-to-end training nature, DeepLabCut represents a significant contribution, although the pose is estimated frame by frame and temporal information is discarded.

Most of the approaches above either require specialized setups or they ignore the temporal properties of motion and try to solve the tracking problem as a frame-localized phenomenon. Here we propose a dense segmentation supervised approach for tracking regions of interest (ROI), limbs in this case, which are moving coherently.

More specifically, the approach is targeted towards limb tracking in situations where the limbs might appear and exhibit motion similar to other body parts, such as the tail. Because of motion and the change of perspective, limbs do not appear rigid throughout the video. Since they deform from frame to frame, traditional object recognition methods do not perform well.

III. MATERIALS AND METHODS

In this section, we provide a brief introduction to the experimental methods and computational concepts we used to build our pipeline.

A. ANIMAL EXPERIMENTS

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. For a complete description of the surgical and behavioral preparations refer to [16], [17]. We used N = 4 males 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 custom-made two-piece headplate [18] was attached to the animal's head. A 3mm or 5mm-wide craniotomy was drilled over the paranormal area of cerebellar lobule VI. After 15 hrs delay for animal recovery the top plate was removed for delivery of AAV1.Syn.GCaMP6f.WPRE.SV40 [Penn Vector Core, lot AV-1-PV2822] virus ([17]). 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 for repeated intervals over 5 days of incremental exposure. After habituation, animals were exposed to a variety of stimuli. Some of the movies were taken from animals that were undergoing eyeblink conditioning [17]. Training consisted of repeated parings of two stimuli, either whisker-puff/eye-puff or light/eye-puff, separated by intervals of 250 or 440ms respectively. This training often induced movements in an otherwise-still mouse. The stimuli consisted of (i) a periorbital airpuff (10-20psi, 30ms in duration, delivered via a plastic needle placed 5mm from the cornea and pointed at it, (ii) a flash of light (400nm, 500ms), or (iii) an airpuff to whisker vibrissae (2-3psi, 800ms).

B. NOTATION AND METHOD OVERVIEW

In table 1 we summarize the notation employed in the rest of the manuscript. As a general rule, we use bold face capital letters to denote tensors, (normal) capital letters to denote scalars, bold face small letters to denote vectors and normal small letters to denote indices. To be consistent, $\mathbf{I}$ denotes the whole video (consisting of T frames with $W \times H$ pixels each), $\mathbf{I}_i$ denotes only the i-th frame of the video, $\mathbf{I}_{i,j}$ denotes j-th superpixel of the i-th frame. We use a window of a fixed size $Ws \times Hs$ per superpixel (see section III-C). We will consistently use i for frame index in a video and j for j-th superpixel in a video frame. If an index appears in small brackets, it corresponds to the exact locations of the matrix elements. For example, $I_i(x_k, y_k)$ corresponds to a pixel (element) in $\mathbf{I}$ at k-th x and k-th y location in frame i.

Figure 1 summarizes the main three steps of the proposed method. Given a video $\mathbf{I}$ consisting of T frames:

1) For each frame, we compute the superpixels of the image. For each superpixel, we look D_{cnn} frames into the past, we find the closest matches for the superpixel in the time axis, and we stack them in chronological order to construct what we name motion tubes $\mathbf{M}_{i,j}$, i.e. a tensor of size $Ws \times Hs \times D_{cnn}$ for each superpixel j of frame i.
2) We extract appearance features from each motion tube using a trained CNN as feature extractor. It will produce a feature vector of length L_f for each motion tube. The CNN parameters are previously learned with a training set containing motion tubes from limb and non limb regions (see III-C).
3) We stack sets of D_{lstm} features extracted from the CNNs to construct the motion sequences $\mathbf{X}_{i,j}$ of size $D_{lstm} \times L_f$. Each motion sequence is used to train a LSTM that performs the image segmentation taking into account the temporal coherence of the motion tubes (see III-D for more details).

TABLE 1. Notation.

Notation	Meaning
D_{cnn}	Depth of CNN input (motion tube)
D_{lstm}	Depth of LSTM sequence
Ws, Hs	width and height of a superpixel window
W, H, T	Width, height and duration of the video
N	Number of superpixels in one video frame
$\mathbf{I}$	Video consisting on T frames (size $W \times H \times T$)
$\mathbf{I}_i$	Frame at the i-th time step (size $W \times H$)
$\mathbf{F}$	Optical flow tensor (size $W \times H \times 2 \times T - 1$)
$\mathbf{F}_i$	Optical flow associated with I_i (size $W \times H \times 2$)
$\mathbf{I}_{i,j}$	j-th superpixel of I_i (size $Ws \times Hs$)
$\mathbf{M}_{i,j}$ (size $Ws \times Hs \times D_{cnn}$)	Motion tube tensor associated with j-th superpixel of I_i
$\mathbf{X}_{i,j}$ (size $D_{lstm} \times L_f$)	Motion sequence associated with j-th superpixel of I_i
L_f	length of feature vector extracted by CNN

In the following subsections, we detail the methods we used to build the motion tubes, extract the feature vectors and obtain the LSTM output for each sequence.

C. MOTION TUBES

Motion tubes are defined as 3D arrays containing pixel clusters with similar motion fields across time. We first reduce the complexity of a frame by grouping all the pixels into superpixels and then track their progression through time. A superpixel refers to a polygonal part of an image, larger than a normal pixel, that is rendered in a similar color and brightness [19]. To overcome the computational cost of over-segmenting a frame into superpixels, we used the efficient SLIC (Simple Linear Iterative Clustering) method proposed by Achanta *et al.* [20].

Superpixels (especially the ones located in mice skin) may have very similar appearance regardless of their state of motion. To effectively use superpixels, we introduce temporal context to them, treating time as the third dimension. In addition, superpixels differ in shape and size. In order to find where a superpixel is located in the next frame (motion path), we propose to use optical flow tracking [21]. Notice that the shape of the superpixels forming the motion tube is deformable. To make the method computationally tractable, we used a fixed sized window to construct the tube ($Ws \times Hs$). In addition, temporal window slices from superpixels of the motion tube (along the depth dimension) may not necessarily refer to the same fixed spatial locations in successive frames, as body parts migrate along the image coordinates. Fig. 2 shows an example of a superpixel tracked backwards through time. The depth of the tube (D_{cnn}) is controlled by the user and can be varied to change the extent of temporal context captured by the tube.

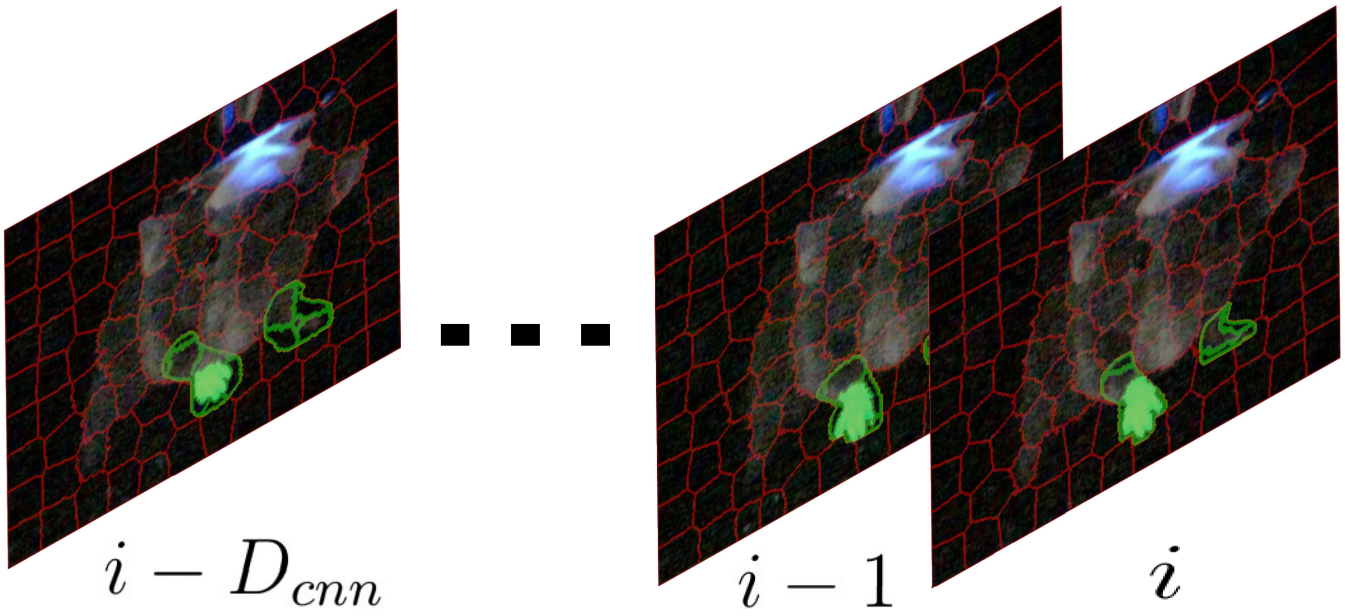


FIGURE 2. **Enlarged view of how the constituent superpixel of a motion tube is tracked backwards through time. The superpixel shaded in green color is forming the tube whose closest relatives are tracked backward in time.**

More formally, let's assume we are dealing with a video with W, H and T dimensions (width, height and number of frames), being $I_i(x_k, y_k)$ each pixel at position x_k and y_k in frame i. SLIC algorithm clusters all these pixels in $\mathbf{I}_i$ into N superpixels. The superpixels generated from current frame are not guaranteed to be present on the same spatial coordinates in the next (or previous) frame. Also, a direct temporal link between superpixels generated from the current frame to superpixels generated in the next (or previous) frame cannot be established. However, the apparent motion of superpixel centroids can give a rough estimate for its closest relatives in the next (or previous) frame if we assume that there aren't any abrupt changes in luminosity, shape and position of the objects in successive frames. We establish a temporal link between the superpixel in the current frame and its closest match in previous or next frame and we stack them up to generate a 3D tube.

To generate the motion tubes of depth D_{cnn} for the N superpixels per each frame $i \in 1 \ldots T$, we proceed as follows:

1) We compute the optical flow from the 3D structure $\mathbf{I}$ representing the video (being $\mathbf{I}_i$ the i-th frame). We store it in a flow tensor $\mathbf{F}$ of size $W \times H \times 2 \times T - 1$. $\mathbf{F}_i$ represents the optical flow associated to frame i. The optical flow refers to the modelling of apparent motion between two successive frames. It was introduced by the American psychologist James J. Gibson in the 1940s [21], and it has been consistently used to model the perception of movement by the observer [22]–[25]. In the field of robotics and computer vision, optical flow is used in image processing and control of navigation including motion detection, object segmentation, time-to-contact information, the focus of expansion calculations, luminance, motion-compensated encoding, and stereo disparity measurement [25]. In our case, the optical flow offers a solution for tracking the temporal behaviour of superpixels in successive frames of a video. We used the Python implementation from openCV [26] of the Lucas-Kannade algorithm [27].
2) We compute the centroid $\mathbf{C}_{i,j}$ of each j-th superpixel (consisting of a subimage $\mathbf{S}_{i,j}$ with a fixed size $Ws \times Hs$) from every frame i.
3) We estimate the coordinates of the centroid on the previous frame $i-1$ using the current coordinates and the optical flow vector.
4) We compute the distance from the estimated centroid $\mathbf{C}_{i-1,j}$ and all the superpixels in the previous frame.
5) We add the window of fixed size $Ws \times Hs$ centered on the closest centroid at frame $i-1$ to the motion tube $\mathbf{M}_{i,j}$.

Notice that the procedure of locating the closest match of superpixels in previous frames allows to handle situations where two superpixels converge in time (typically because the limb regions shrink as movement dynamics evolve). The algorithm 1 details the main steps of the procedure that outputs N motion tubes for each frame.

To reduce the dimensionality of the data and to extract meaningful appearance features, we use a 4-layer Convolutional Neural Network (CNN) [28]. We conjecture that since we train the CNN to learn how to discriminate between limb/non limb categories from motion tubes, the features extracted will be more descriptive than hand crafted ones. The input to the first convolutional layer is a tensor with the

Algorithm 1 Motion Tubes Construction

```
Result: Motion tubes for the superpixel centered at $\mathbf{C}_{i,j}$
. $k = 0$; M = { };
if $i < D_{cnn}$ then
                                        ▷ Not traversed the minimum
    number of frames yet while Number of available frames
    is less than required ($i < D_{cnn}$) do
        Append $\mathbf{I}_i$ to $\mathbf{I}$ at position $i$;
        Append $\mathbf{F}_i$ to $\mathbf{F}$ at position $i$;
    end
else
    Start from 1st superpixel ($j = 0$)
    while we have available superpixels ($j < N$) do
        Start from current frame i ($k = 0$)
        while we haven't traversed backwards $D_{cnn}$ frames
        ($k < D_{cnn}$) do
            if current frame i (k==0) then
                Place $\mathbf{S}^s_{i,j}$ at last location of the tube
            else
                Find centroid $\mathbf{C}_{i,j}$ of superpixel $\mathbf{S}_{i,j}$ and store
                it in $(x_{sj}, y_{sj})$.

                Project it onto frame $\mathbf{I}_{i-k}$ using the
                optical flow vector as $(x_{sj*}, y_{sj*}) =$
                $(x_{sj}, y_{sj}) + \mathbf{F}_{i-k}(x_{sj}, y_{sj})$.

                Find distance between the centroid of
                projected superpixel and the centroids of all
                superpixels in previous frame and store it in
                $D_{s:}$. The closest relative of superpixel with
                centroid $(x_j, y_j)$ is the one returned by
                $C^s_{i,j*} = argmin(D_{s:})$.

                Append the superpixel $\mathbf{S}_{i,j*}$ at location $D_{cnn} -$
                $k$ of motion tube $\mathbf{M}_{i-k,j}$.
            end
            $k = k + 1$;
        end
        $j = j + 1$
    end
end
```

motion tube $\mathbf{M}$ of size $Ws \times Hs \times D_{cnn}$ where Ws and Hs represents spatial dimensions while D_{cnm} represents the tube depth. The output of the CNN is a binary value (limb/non limb) for each input motion tube. Nevertheless, we extract as a feature vector $\mathbf{v}_1$ the values obtained from the last fully connected layer, of length L_f. Particularly we used 64 values (neurons at the last fully connected layer) in our experiments. Table 2 details the parameters for each layer used in this paper.

D. MOTION SEQUENCES

Motion sequences integrate several temporally consecutive motion tubes to learn a sequential classifier that performs the image segmentation for limb tracking. For a given superpixel j in frame i, we extract a one dimensional motion tube vector $\mathbf{v}_1$. Then we do the same process for the closest match of this superpixel in the previous frame and extract another one dimensional feature vector $\mathbf{x}_2$. We repeat the process for the previous D_{lstm} frames. Finally we stack all these one dimensional feature vectors to form the motion sequence $\mathbf{V} = [\mathbf{v}_1; \mathbf{v}_2; \ldots; \mathbf{v}_{D_{lstm}}]$ of size $D_{lstm} \times L_f$. We compute one motion sequence per superpixel and we train a LSTM Neural Network [29] to predict the probability of the sequence of belonging to a limb.

TABLE 2. Parameter values for the layers.

	Layer 1	Layer 2	Layer 3	Layer 4
Number of filters	8	16	32	64
Filter size	3	3	3	3
Stride of pooling	2	2	2	2

E. TRAINING DATA FOR THE CNN AND LSTM REGRESSORS

We use a CNN for feature extraction and a LSTM for image segmentation. In both cases we train these models to provide a probability of the superpixel of being limb /non limb. The training data is generated by manually annotating limbs in video frames. The limbs (moving or still) are labeled accurately by tracing their boundaries and then extracting a mask.

We use a mean squared error loss for the regression task. Nevertheless, training data are annotated in binary terms (limbs / non limb) at a pixel level (segmentation mask). To provide a continuous value for all the pixels contained in the j-th superpixel $\mathbf{I}_{i,j}(x, y)$, the regression target used is proportional to the intersection of the superpixel and the annotated segmentation mask. We first select all the x_j, y_j pixels belonging to the superpixel j, and assign a regression target value to that superpixel using eq.1.

$$Y_{j*} = \frac{\{(x, y)|x \in x_j, y \in y_j \wedge I(x_j, y_j) = 1\}}{\{(x, y)|x \in x_j, y \in y_j\}} \tag{1}$$

The masks of superpixels with $Y_{j*} \leq 0.1$ are clipped to 0 (non limb). This 'limb-ness' score is then associated to every pixel within such superpixel j. These new pixel values are used to generate a dense segmentation frame with the same width and height as the original frame. A simple thresholding on such segmentation frame produces a mask. Such mask can be compared to the manually annotated ground truth for evaluation.

IV. DATA SET AND EXPERIMENTS

A. DATA SET

The pipeline is tested on two types of video data, frontal and lateral videos (camera located at the right side of the mice). Both sets of videos are acquired at 120 frames per second and have a resolution of 240×320 pixels. Each of the frontal videos is 239 frames long while each of the lateral videos is

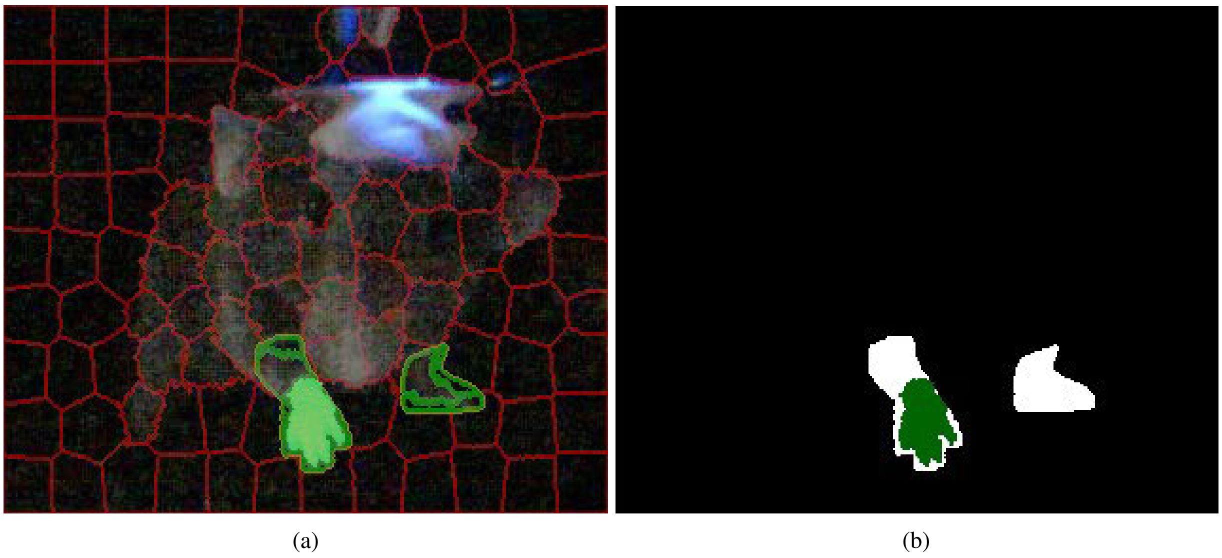


FIGURE 3. **Example of how a label is given to a superpixel. First, the original frame is over-segmented into superpixels. To find out the target label for a superpixel, shaded in green in (a), the corresponding superpixel, also shaded in green in (b) is extracted and then evaluated according to Eq. 1.**

767 frames long. The frontal videos have a high amount of salt and pepper noise as well as equipment noise while the lateral videos are relatively noise free. In order to evaluate the quality of the videos, we calculated the average value of SNR (signal-to-noise-ratio) across all video frames (4.8db and 18.db for the frontal and lateral cases respectively). Typical frames from both lateral and frontal annotated videos are shown in Fig. 4a and 4c.

Three frontal video (717 frames) and two lateral videos (1534 frames) were annotated by three human annotators independently. The degree of agreement between the three human annotators is higher than 95% on average, therefore the annotation is reliable to be used for training and testing purposes.

B. PIPELINE DETAILS

1) MOTION TUBES GENERATION

We have experimented with a tube depth of 9 frames (75 ms). The size of the superpixels is controlled by N (number of superpixels in a frame). To construct the motion tubes, we extracted a 61×61 patch centered on each superpixel and we resized it to 41×41 pixels. The tubes can only be generated once we have traversed at least D_{cnn} frames of the video, so with a video length of 239 frames, $D_{cnn} = 9$, $N = 100$ we will have a total of 230×100 tubes, and the resulting data will have dimensions $41 \times 41 \times 9 \times 23000$.

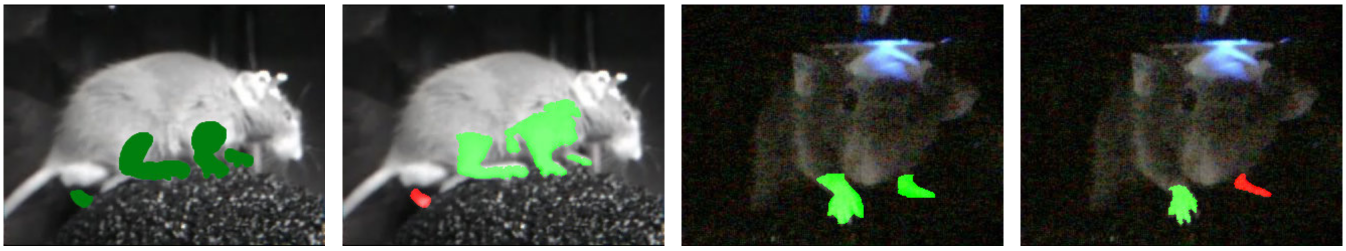
2) BUILDING MOTION SEQUENCES FROM FEATURES EXTRACTED BY CNN

We trained a CNN on the motion tubes with an input layer, 4 hybrid layers, a dropout layer (ratio = 0.5), a fully connected layer with 64 elements and a regression layer. The hybrid layers consist of a convolutional layer, a reLu layer, a batch normalization layer and an average pooling layer with following parameter values.

Once trained, we used the CNN as a feature extractor. We applied it to each motion tube and kept only the 64 features extracted by the fully connected layer. These features contain appearance and temporal information from each superpixel.

3) LIMBS DETECTION BY LSTM

The features generated by the CNN are used to create motion sequences for the LSTMs with $D_{lstm} = 15$ (15 features-vectors-deep sequences). Hence, a superpixel is represented by 15 spatio-temporal feature vectors of length 64 each ($L_f = 64$) stacked together in a chronologically ordered temporal sequence. So for a video of 239 frames, with a motion tube depth of 9 frames and motion sequence depth of 15 frames, the first motion sequence will be available after 24 frames which will leave $239 - 24 = 215$ frames. With 100 superpixels in each frame, feature length of 64 and motion sequence depth of 15, we will end up with a training data of $15 \times 64 \times 21500$ motion sequences. We trained an LSTM with 6 layers (input, three layers with 64, 32, and 16 cells, a fully connected and a regression layer). The input to the LSTM is a motion sequence $\mathbf{M}_{i,j}$ (generated by each superpixel j in frame i) and the output is the amount of limb-ness (a value between 0 and 1, describing our confidence on a given superpixel to belong to a limb) calculated by equation 1. To obtain a binary mask, we simply apply a threshold (0.5). It is to be noted that at the training stage, we clipped all the target values below 0.1 to 0, but we keep the clipping threshold at 0.5 at inference stage to reduce the probability of false positives. All the superpixels with a limb-ness value greater than this threshold are considered to be parts of limbs.



(a) A sample frame with limbs annotated from a lateral video. (b) Detection results overlaid on original frame at inference time. (c) A sample frame with limbs annotated from a frontal video. (d) Predicted limbs overlaid on a sample frame of the frontal video.

FIGURE 4. A sample frame with limbs annotated from a lateral video.

C. EXPERIMENTS

To evaluate the effectiveness of proposed pipeline, we performed the experiments on both frontal videos and lateral videos in three settings:

1) SETTING 1

We trained the proposed pipeline on one video (239 training frames for frontal videos and 767 training frames for lateral videos) and tested it on the other video (2 videos in frontal settings with 478 frames and one video in lateral setting with 767 frames).

2) SETTING 2

We trained the proposed pipeline on 30% of the available frames and tested it on remaining 70% frames. This translates to 217 training frames for frontal videos (239 frames per video, 3 videos, 717 total frames, 217 for training and 500 for testing) and 460 training frames for lateral videos (767 frames per video, 2 videos, 1534 total frames, 460 for training and 1074 for testing).

3) SETTING 3

In setting 3, we trained the proposed pipeline on 10% of the available frames and tested it on remaining 90% frames. This translates to 72 training frames for frontal videos (239 frames per video, 3 videos, 717 total frames, 72 frames for training and 645 for testing) and 154 training frames for lateral videos (767 frames per video, 2 videos, 1534 total frames, 154 for training and 1380 for testing).

V. RESULTS

We conducted both qualitative and quantitative analysis of the proposed pipeline. For qualitative analysis, we superimposed detection results for limbs onto the original videos. The videos can be found in supplementary and multimedia data associated with the article. For quantitative analysis, we calculated the precision $(TP/(TP+FP))$, recall $(TP)/(TP+FN)$, detection accuracy $(TP + TN)/(TP + TN + FP + FN)$ and Jaccard Index [30]. In frame I_i, the limb is represented by mask I_i^L and the detected limb is represented by mask $I_i^{L^*}$,

TABLE 3. Performance for lateral videos in settings 1-3.

Setting 1	Precision	Recall	Accuracy
Front right limb	98.0%	98.5%	96.6%
Hind right limb	96.8%	98.3%	95.3%
Setting 2			
Front right limb	96.6%	91.3%	88.5%
Hind right limb	97.6%	96.2%	95.0%
Setting 3			
Front right limb	97.8%	94.8%	92.8%
Hind right limb	94.7%	95.0%	90.1%

then the Jaccard Index is defined as:

$$J_i = \frac{I_i^L \cap I_i^{L^*}}{I_i^L \cup I_i^{L^*}} \tag{2}$$

If the Jaccard Index is higher than a threshold (0.5 for all the settings in this paper), we conclude we have detected the limb, and vice versa.

A. LATERAL VIDEOS

A sample of an annotated frame from lateral videos is depicted in Fig. 4a. The corresponding detection results are overlaid on a sample frame in Fig. 4b. In order to understand the impact of the number of frames used for training the pipeline, we conducted experiments with lateral videos in the three settings described above. Particularly, in setting one, we used one video for training and one video for testing (767 frames for training and 767 frames for testing or 6.3 seconds of video for training and 6.3 seconds for testing). In the second setting, we used 30% frames for training and 70% frames for testing (460 frames for training and 1074 frames for testing or 3.8 seconds of video for training and 8.9 seconds for testing). In the third setting, we used 10% frames for training and 90% frames for testing (154 frames for training and 1380 frames for testing or 1.3 seconds of video for training and 11.5 seconds for testing). The results for these three settings are summarized in table 3.

B. FRONTAL VIDEOS

Compared to the lateral videos, frontal videos contain a higher amount of noise (both salt and pepper and equipment

TABLE 4. Performance for frontal videos in settings 1-3.

Setting 1	Precision	Recall	Accuracy
Front left limb	91.5%	71.4 %	79.1%
Front right limb	92.9%	91.7%	88.2%
Hind right limb	89.1%	70.7%	80.5%
Setting 2			
Front left limb	95.7%	71.0%	77.8%
Front right limb	93.8%	87.9%	86.3%
Hind right limb	92.1%	72.5%	82.2%
Setting 3			
Front left limb	94.3%	68.3%	74.3%
Front right limb	93.4%	87.6%	85.1%
Hind right limb	91.8%	67.4%	78.0%

noise) and present more occlusion of the limbs. Therefore, limbs are not always identifiable from the body or background. For this reason, the detection and tracking performance are worse than in the case of lateral videos. A sample annotated frame is shown in Fig. 4c while Fig. 4d shows the detection results for a predicted segmentation overlaid on the original frame. The quantitative results for settings 1, 2 and 3 (as described above) are shown in table 4.

As evident from tables 3, and 4, the detection performance does drop as we reduce the number of training samples. However, the drop in performance is moderate with respect to the reduction in the number of samples (A maximum drop of 8.1% in accuracy for lateral videos while a maximum drop of 4.2% in accuracy for frontal videos against a 60% drop of number of training samples).

C. TRACKING PRECISION EVALUATION

To evaluate the tracking performance of the proposed pipeline with respect to state-of-the-art methods, we computed the centroids of the front left limb in frontal videos and hind right limbs in lateral videos in both manually annotated (Ground Truth) and predicted frames. We compared them qualitatively by plotting the annotated and actual tracks side by side and then compared them quantitatively by finding the mean distance between the manually annotated and the predicted positions (centroids) according to equation 3.

$$D_i^p = \sqrt{(x_G^c - x_P^c)^2 + (x_G^c - y_P^c)^2} \quad (3)$$

where x_G^c and y_G^c are the coordinates in the ground truth frames and x_P^c and y_P^c are the coordinates in the predicted frames. We found that the manually annotated and predicted limbs are separated on average by 3.84 pixels in frontal videos and 2.24 pixels in lateral videos. We compared the lateral video results with the Haar cascades method defined in [16]. The separation grows to 9.5 pixels.

We also compared the performance of our approach with more accurate approaches: the Deeplabcut method proposed in [31] and the DeepPoseKit software toolkit [32]. This recent open software addresses the speed and robustness problems in animal pose estimation tasks. Authors employ two efficient multi-scale deep-learning models,

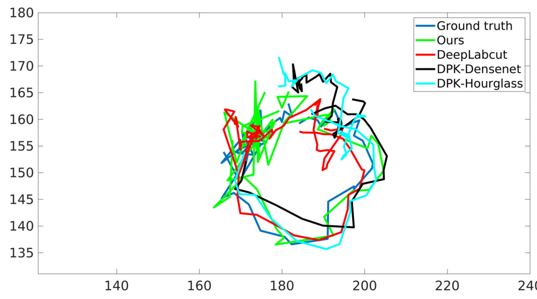


FIGURE 5. Actual path and path predicted by our approach and DeepLabcut [31] of front left limb in a frontal video. The path is calculated by finding centroids of the front left limb in each frame and then plotting its y coordinate against its x coordinate.

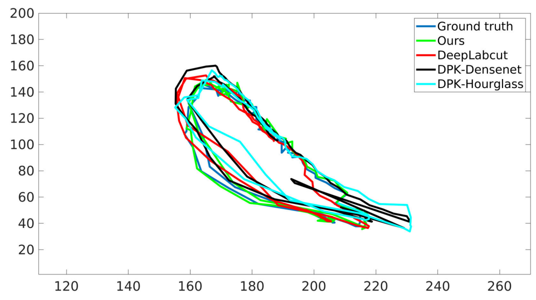


FIGURE 6. Actual path and path predicted by our approach and DeepLabcut [31] of hind right limb in a lateral video. The path is calculated by finding centroids of hind right limb in each frame and then plotting its y coordinate against its x coordinate.

TABLE 5. Tracking mean squared error of the trajectories in the frontal and lateral settings.

Method	Frontal	Lateral
Deeplabcut	5.5	6.35
Deepposekit (StackedHourglass)	3.97	5.39
Deepposekit (StackedDensenet)	4.74	5.63
Motion Tubes	2.24	3.84

called Stacked DenseNet and Stacked Hourglass, and a fast GPU-based peak-detection algorithm for estimating keypoint locations with subpixel precision. Authors report similar level of accuracy as of the contemporary methods with a significant speedup. To make the comparison fair, we trained the Deeplabcut and DeepPoseKit on the same videos. The mean distance between the Deeplabcut obtained coordinates and the ground truth centroids was 5.5 pixels in the frontal videos and 6.35 pixels in the lateral videos. Similarly, the best performing DeepPoseKit instance (Stacked Hourglass) obtained 3.97 and 5.39 mean distance respectively. In Figures 5 and 6 we plot the limb tracks in both experiments, and Table 5 summarizes the results of the tracking precision experiments.

VI. CONCLUSION AND FUTURE WORK

We have proposed a deep learning-based solution to annotate behavioral data. In a typical behavioral neuroscience data set, researchers aim to identify limbs in every frame. Conventional techniques rely on a frame by frame image segmentation or object detection. Our approach is based on the notion that limbs are regions of the frames which feature specific and learnable spatio-temporal characteristics. We defined the notion of motion tubes and motion sequences, that use compact representations (superpixels) to simultaneously extract appearance and temporal features on videos. We used features learned by training a CNN for a segmentation task instead of hand crafted approaches. We obtained promising results on two different acquisition conditions (lateral and frontal videos) and under different noise patterns.

We have developed this approach under the assumption that the nimal, in this case the mouse, is head-fixed. Therefore, for our approach to work in freely moving animals, additional steps need to be included.

To further enhance the effectiveness of the proposed approach, future work can be undertaken in the following directions:

- Building of motion tubes by a locally trained network instead of using optical flow.
- Perform Tube depth and width parameter optimization.

AUTHORS CONTRIBUTIONS

Conceptualization, W.A. and D.M.; methodology, W.A. and D.M.; software, W.A.; formal analysis, W.A. and D.M.; investigation, W.A. and D.M.; resources, W.A., D.M and A.G.; data acquisition, A.G; data curation, W.A.; writing–original draft preparation, W.A., D.M and A.G.; writing–review and editing, W.A., D.M and A.G.; visualization, W.A.; supervision, D.M.; project administration, D.M.; funding acquisition, D.M.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] A. Mathis, P. Mamidanna, T. Abe, K. M. Cury, V. N. Murthy, M. W. Mathis, and M. Bethge, "Markerless tracking of user-defined features with deep learning," 2018, *arXiv:1804.03142*. [Online]. Available: http://arxiv.org/abs/1804.03142

[2] W. Abbas and D. Masip Rodo, "Computer methods for automatic locomotion and gesture tracking in mice and small animals for neuroscience applications: A survey," *Sensors*, vol. 19, no. 15, p. 3274, Jul. 2019.

[3] J. Kain, C. Stokes, Q. Gaudry, X. Song, J. Foley, R. Wilson, and B. de Bivort, "Leg-tracking and automated behavioural classification in drosophila," *Nature Commun.*, vol. 4, no. 1, p. 1910, May 2013.

[4] S. Roy, J. L. Bryant, Y. Cao, and D. H. Heck, "High-precision, three-dimensional tracking of mouse whisker movements with optical motion capture technology," *Frontiers Behav. Neurosci.*, vol. 5, p. 27, 2011.

[5] S. Tashman and W. Anderst, "in-vivo measurement of dynamic joint motion using high speed biplane radiography and CT: Application to canine ACL deficiency," *J. Biomechanical Eng.*, vol. 125, no. 2, pp. 238–245, Apr. 2003.

[6] H. P. M. A. HarveyRoberto Bermejo, "Discriminative whisking in the head-fixed rat: Optoelectronic monitoring during tactile detection and discrimination tasks," *Somatosensory Motor Res.*, vol. 18, no. 3, pp. 211–222, Jul. 2009.

[7] A. Kyme, S. Meikle, C. Baldock, and R. Fulton, "Tracking and characterizing the head motion of unanaesthetized rats in positron emission tomography," *J. Roy. Soc. Interface*, vol. 9, no. 76, pp. 3094–3107, Jun. 2012.

[8] A. Z. Kyme, V. W. Zhou, S. R. Meikle, and R. R. Fulton, "Real-time 3D motion tracking for small animal brain PET," *Phys. Med. Biol.*, vol. 53, no. 10, pp. 2651–2666, Apr. 2008.

[9] A. Z. Kyme, V. W. Zhou, S. R. Meikle, C. Baldock, and R. R. Fulton, "Optimised motion tracking for positron emission tomography studies of brain function in awake rats," *PLoS ONE*, vol. 6, no. 7, Jul. 2011, Art. no. e21727.

[10] M. O. Pasquet, M. Tihy, A. Gourgeon, M. N. Pompili, B. P. Godsil, C. Léna, and G. P. Dugué, "Wireless inertial measurement of head kinematics in freely-moving rats," *Sci. Rep.*, vol. 6, no. 1, Oct. 2016, Art. no. 35689.

[11] Z. Wang, S. A. Mirbozorgi, and M. Ghovanloo, "Towards a kinect-based behavior recognition and analysis system for small animals," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2015, pp. 1–4.

[12] J. P. Monteiro, H. P. Oliveira, P. Aguiar, and J. S. Cardoso, "A depth-map approach for automatic mice behavior recognition," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 2261–2265.

[13] T. Palmér, K. Åström, O. Enqvist, N. Ivica, and P. Petersson, "Rat paw tracking for detailed motion analysis," in *Proc. Vis. Observ. Anal. Vertebrate Insect Behav.*, 2014.

[14] T. Palmér, M. Tamtè, P. Halje, O. Enqvist, and P. Petersson, "A system for automated tracking of motor components in neurophysiological research," *J. Neurosci. Methods*, vol. 205, no. 2, pp. 334–344, Apr. 2012.

[15] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele, "DeepCut: Joint subset partition and labeling for multi person pose estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4929–4937.

[16] A. Giovannucci, E. A. Pnevmatikakis, B. Deverett, T. Pereira, J. Fondriest, M. J. Brady, S. S.-H. Wang, W. Abbas, P. Parés, and D. Masip, "Automated gesture tracking in head-fixed mice," *J. Neurosci. Methods*, vol. 300, pp. 184–195, Apr. 2018.

[17] A. Giovannucci, A. Badura, B. Deverett, F. Najafi, T. D. Pereira, Z. Gao, I. Ozden, A. D. Kloth, E. Pnevmatikakis, L. Paninski, C. I. De Zeeuw, J. F. Medina, and S. S.-H. Wang, "Cerebellar granule cells acquire a widespread predictive feedback signal during motor learning," *Nature Neurosci.*, vol. 20, no. 5, pp. 727–734, Mar. 2017.

[18] D. A. Dombeck, C. D. Harvey, L. Tian, L. L. Looger, and D. W. Tank, "Functional imaging of hippocampal place cells at cellular resolution during virtual navigation," *Nature Neurosci.*, vol. 13, no. 11, pp. 1433–1440, Oct. 2010.

[19] *Superpixel*. Accessed: Sep. 26, 2019. [Online]. Available: https://www.yourdictionary.com/superpixel

[20] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[21] K. Nakayama, "James J. Gibson: An appreciation.," *Psychol. Rev.*, vol. 101, no. 2, pp. 329–335, 1994.

[22] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, nos. 1–3, pp. 185–203, Aug. 1981.

[23] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43–77, Feb. 1994.

[24] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2004, pp. 25–36.

[25] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, Sep. 1995.

[26] G. Bradski, "The OpenCV Library," *Dr. Dobb's J. Softw. Tools*, 2000.

[27] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 2. San Francisco, CA, USA: Morgan Kaufmann, 1981, pp. 674–679. [Online]. Available: http://dl.acm.org/citation.cfm?id=1623264.1623280

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] P.-N. Tan, *Introduction to Data Mining*. London, U.K.: Pearson, 2018.

[31] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: Markerless pose estimation of user-defined body parts with deep learning," *Nature Neurosci.*, vol. 21, Sep. 2018, Art. no. 1281.

[32] J. M. Graving, D. Chae, H. Naik, L. Li, B. Koger, B. R. Costelloe, and I. D. Couzin, "Fast and robust animal pose estimation," *bioRxiv*, Apr. 2019, Art. no. 620245.

WASEEM ABBAS received the master's degree from the College of Ocean System Engineering, Jeju National University, South Korea, in 2016. He is currently pursuing the Ph.D. degree with the Computer Science Multimedia and Telecommunication Department, Universitat Oberta de Catalunya, Barcelona, Spain. He worked on statistical signal processing in maritime environment and pattern recognition/machine learning in acoustic fingerprinting at Jeju National University. He has a background in electrical engineering and data analysis. He is also working under the supervision of Dr. D. Masip on gesture and neural data analysis with deep learning methods for neuroscience applications under the umbrella of the Scene Understanding and Artificial Intelligence (SUNAI) research group. He is also working on an end-to-end training pipeline for neural activity detection in two photon calcium imaging data.

DAVID MASIP (Senior Member, IEEE) received the degree in computer science from the Universitat Autonoma de Barcelona, and the Ph.D. degree in September 2005. He worked as an Assistant Professor at the Applied Mathematics Department, Universitat de Barcelona. He has been a Professor with the Computer Science Multimedia and Telecommunication Department, Universitat Oberta de Catalunya, since February 2007, and has also been the Director of the UOC Doctoral School, since 2015. He performed a postdoctoral stay at Princeton University, in 2013. He is currently the Director of the Scene Understanding and Artificial Intelligence (SUNAI) research group. His main research interests are computer vision, deep learning algorithms, animal behavior analysis, and facial expression classification. He received the FPI Grant, in 2001, for starting his Ph.D. degree at the Computer Vision Center, Spain.

ANDREA GIOVANNUCCI received the degree in associative learning in the cerebellum using multiphoton imaging from the Princeton Neuroscience Institute. During his Ph.D. at the Autonoma University of Barcelona and the IIIA-CSIC, he developed scalable algorithms and mechanisms to engineer the organization of multiagent systems. He was a Machine Learning Data Scientist with the Flatiron Institute, Simons Foundation, and a Postdoctoral Fellow of experimental neuroscience with the Princeton Neuroscience Institute. He is currently an Assistant Professor of neural engineering with the Joint Department of Biomedical Engineering, University of North Carolina at Chapel Hill/North Carolina State University (UNC/NCSU), where he develops experimental and computational tools to study the brain. His research at the Flatiron Institute has focused on developing algorithms and tools for the analysis of large imaging datasets and animal behavior. This effort has culminated in an open-source software suite—CaImAn—that is widely employed by the neuroscience community and includes algorithms to solve several preprocessing problems.