Video hardware and software for the International Brain Laboratory

Dan Birman[5] , Niccolò Bonacchi[1], Kelly Buchanan[5], Gaëlle Chapuis[2], Julia M. Huntenburg[6], Guido Meijer[1], Liam Paninski[5], Michael Schartner[1], Karel Svoboda[4], Miles Wells[3], Matthew R. Whiteway[5], Olivier Winter[1], The International Brain Laboratory

[1]Champalimaud Centre for the Unknown, Lisbon, Portugal; [2]University of Geneva, Switzerland; [3]University College London, United Kingdom; [4]Allen Institute, Seattle, USA; [5]Mortimer B. Zuckerman Mind Brain Behavior Institute and Department of Neuroscience, Columbia University, New York, USA; [6]Max Planck Institute for Biological Cybernetics, Tuebingen, Germany

Table of content

Summary

Neural dynamics control movements, and movements in turn produce sensory input that modulate neural dynamics. It is therefore important to monitor movements in great detail in order to interpret neural dynamics. This monitoring can be achieved in a simple and inexpensive manner through the use of video cameras. This document describes the current hardware and software for the video analysis pipeline in the IBL. In the IBL recording rig, we use three video cameras to film the mouse from the left and right side, as well as from above (body camera). We use DeepLabCut[1], a toolbox for markerless pose estimation based on deep learning, to track different points in the videos recorded during the IBL task. With the three-camera setup we routinely track pupils, paws, tongue and nose bilaterally with a test error of fewer than 5 pixels (1 px corresponding to approximately 0.05 mm; image dimensions 1280 x 1024 px for one side camera). Furthermore, tracking was highly robust to mp4 compression (h.264). However, achieving robust tracking across many hundreds of heterogeneous videos acquired across a dozen experimental labs required significant semi-automated quality control and training set curation efforts, including manual labeling of tens of thousands of images.

Hardware

One side camera in training rigs

The rig in which mice are trained on the IBL task is equipped with a single Chameleon3 camera with a 16 mm fixed zoom lens (Figure 1a, see Table 2 for list of hardware components). The camera captures a single side view at 30 Hz (Figure 1b) at full camera resolution (1280 x 1024 px) for tracking the pupil, at least one paw, and the tongue. For the latter it is important that the water spout and the mouse's mouth are not too close to each other, to ensure that the tongue is clearly visible during licking. Imaging is performed through an IR longpass filter (730 nm). Lighting is provided by an array of LEDs (850 nm), diffused by an opaque plastic sheet.

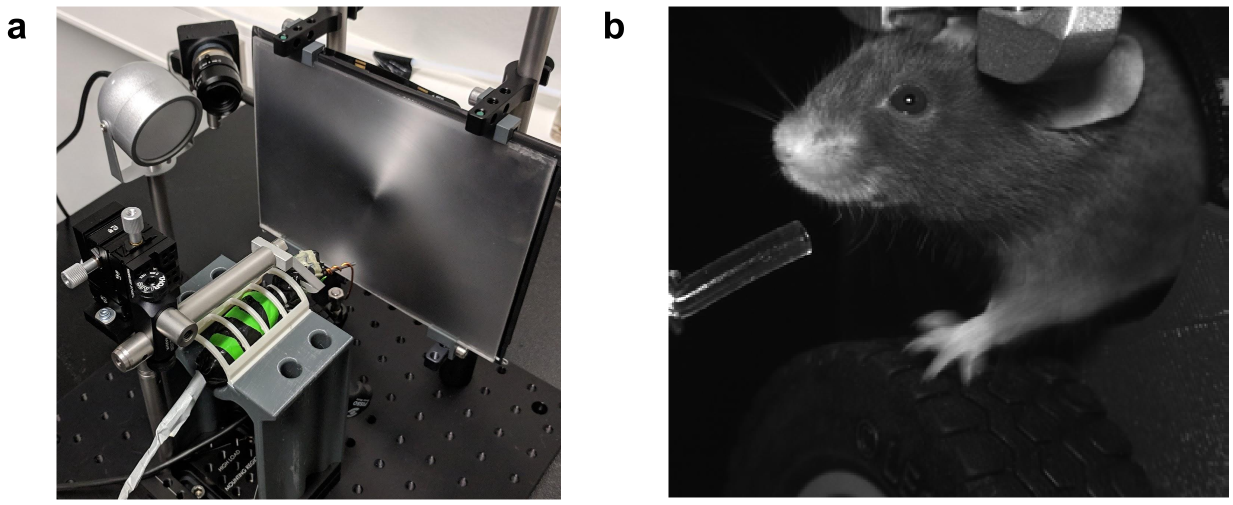


Figure 1. a) Positioning of the camera in the training rig. **b)** Example video of the training rig camera.

Integration with the behavior control hardware (Bpod) is achieved by connecting the pins of the camera to an ethernet port of the Bpod. The camera outputs a 3 Volt (V) TTL pulse during every frame capture, which is logged and timestamped by the Bpod.

Within the IBL it is extremely important that the images of cameras from different rigs are comparable. Therefore, a program was created which allows users to set up and position the side camera in a standardized way. In the program, the user is presented with an outline of an empty training rig, overlayed with the live video feed of the camera. The user should then change the camera angle by hand until the outline and the video image match (Figure 2a,b).

There are two variables, however, which have to be set physically on the lens of the camera: the focus and the aperture. To do this, the user is instructed to place a mouse in the setup and focus the camera until the eye of the mouse is in focus. The aperture is set by looking at a window in the program which shows all saturated pixels in the live video feed (Figure 2c). The user is then instructed to open the aperture until slight saturation of parts of the mouse starts to appear.

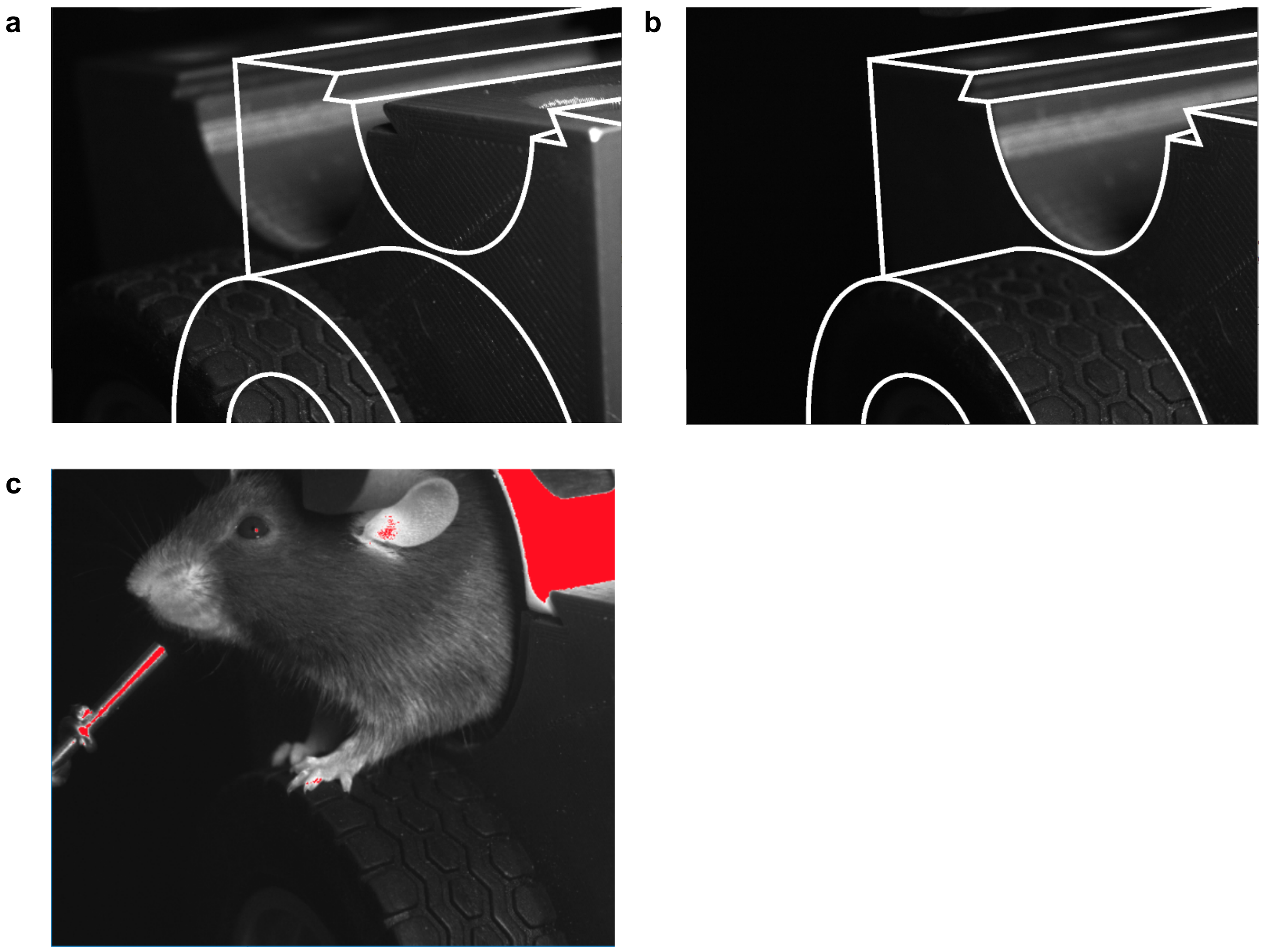


Figure 2. a) The camera image of an empty training rig overlayed with a drawing of another rig. The image and the drawing don't match. **b)** After correctly positioning the camera the line drawing and the video feed match. **c)** The aperture of the lens is slowly opened until slight saturation (red pixels) appear on the mouse.

Two side and one body camera in the recording rigs

After successful training, the mice are transferred to a different rig, in which the electrophysiological recordings during task performance take place. This recording rig is equipped with three cameras: two side cameras and one body camera (Figure 3). Table 1 lists the temporal and spatial resolution of the cameras and other parameters that are set as constant. More details about the recording rig can be found here. The three cameras are connected to a dedicated video capture computer. Bonsai[2] is used as a recording software, which allows for a video stream window to be displayed to the user during the recording.

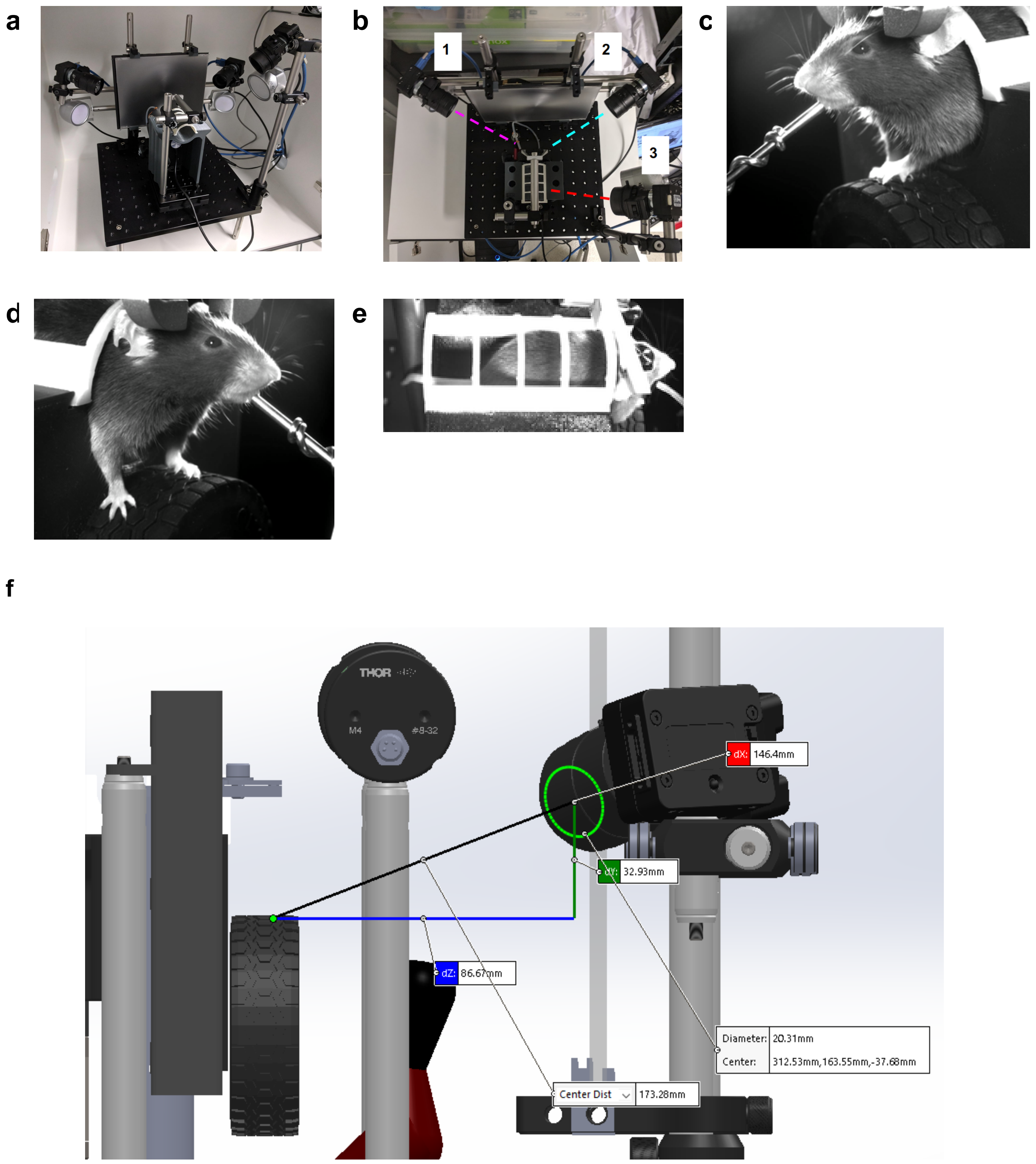


Figure 3. a) Camera setup of the recording rig. **b)** Two cameras track both sides of the mouse (camera 1, 2). One camera records movements of the trunk and tail of the mouse (camera 3). Each camera has one IR-light source. **c)** Example video recorded with the left camera. **d)** Example video recorded with the right camera. **e)** Example video recorded with the body camera. **f)** Positioning of the right camera relative to the target. The left camera is symmetric to this.

camera name	resolution [px]	frame rate [Hz]	exposure [EM]	gain [dB]	shutter time [ms]
left (training rig)	1280 x 1024	30	0.6	12	6
left	1280 x 1024	60	0.6	12	6
right	640 x 512	150	0.6	12	6
body	640 x 512	30	0.6	12	6

Table 1. Camera settings of the one side camera in the training rig and the 3 cameras (two side, one body) in the recording rig.

The frame acquisition jitter is smaller than 1 ms, here tested for setting the side camera to 60 Hz and 150 Hz (Figure 4).

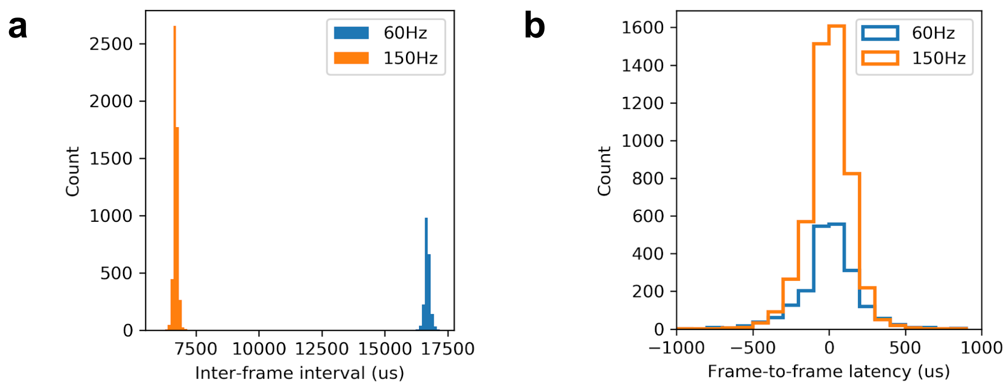


Figure 4. a) Histograms of the inter-frame interval of a three-minute video from two cameras, one rrecording at 60Hz (blue) and the other at 150Hz (orange). There is a certain variation in the latency between successive video frames. **b)** a histogram of frame-to-frame latency of the 60 Hz (blue) and 150 Hz (orange) cameras. Jitter is defined as the standard deviation of the frame-to-frame latency variation. 60 Hz camera jitter: 195.6 μs; 150 Hz camera jitter: 143.2 μs.

The right 150 Hz camera serves as a secondary synchronization source between the electrophysiology acquisition FPGA DAQ and the Bpod (note that there is no FPGA in the training rigs). The right camera sends TTL pulses at every frame capture to both the Bpod and the FPGA. The other two cameras (left and body) send a TTL pulse only to the FPGA. The Bpod sends a synchronization pulse to the FPGA at each trial onset.

The software that writes the frames to disk is not fully initialized before the first frames of the session are sent by the camera. Therefore the first several frames are missing from the video, while the camera TTLs are already recorded by the FPGA and Bpod. In addition, frames are occasionally dropped throughout the session. To account for the resulting mismatch between the video frames and the camera TTLs, some post-hoc changes were made to the setup. Namely, the audio TTLs sent from the sound card were connected to the general purpose IO (GPIO) pin of the camera, which embeds the pin state into the frame data, along with the camera's internal frame count. For newer sessions with this GPIO data, the camera timestamp alignment is done by aligning the GPIO data to the audio TTLs. The camera TTLs that do not correspond to any frame in the video file can then be removed. For sessions without the GPIO data, it is assumed that all missed frames occur at the start (because the video writer is not initialized): the first few camera

TTL timestamps are dropped so that the array matches the number of video frames.

Hardware component list

We used the following video acquisition pc and cameras:

QTY	Product	Specifications
3	High speed camera	CM3-U3-13Y3M-CS
3	Lens	Thorlabs MVL16M23
3	Tripod Adaptors	ACC-01-0003
3	USB3 cable	ACC-01-2300
1	IR pass filter	http://www.leefilters.com/index.php/camera-directory/camera-dir-list/category/infrared
1	Acrylic sheet for diffusing IR illumination	160 mm x 200 mm x 3mm
1	IR reflective film	PR70 (170 mm x 210 mm)
1	PC for video acquisition	Motherboard: MSI X470 GAMING PLUS (sku: 4719072564278) RAM: Crucial Kit 32GB (4 x 8GB) DDR4 2666MHz Ballistix Tactical (sku: BLT4C8G4D26AFTA) CPU: AMD RYZEN 5 1600 3.2GHz - 3.6GHz 16MB AM4 (sku: YD1600BBAEBOX) Graphics card: ASUS GeForce GTX 1050 TI DC2 OC 4GB GD5 (sku: 90YV0A32-M0NA00) SSD (data): Samsung 960 Pro 2TB M.2 NVME (sku: MZ-V6P2T0BW) Power supply: Corsair Modular CS-750W 80 PLUS GOLD (sku: CP-9020078-EU) SSD (OS): Samsung 850 Evo 250GB SATAIII (sku: MZ-75E250B/EU) Case: Fractal Design Define R5 Black Mid Tower Case (sku: FD-CA-DEF-R5-BK) OS: Ubuntu 16.04

Table 2. Video-specific hardware list.

Software

Current DeepLabCut pipeline

We use DeepLabCut[1] (DLC), a toolbox for deep learning-based markerless pose estimation, to automatically track a set of points in the videos recorded during the IBL task. The Python scripts for the current pipeline can be found at https://github.com/int-brain-lab/iblvideo. For the left and right side videos, we track four points surrounding the visible pupil (top, bottom, left, right), two points on the drinking spout (top, bottom), two points on the tip of the tongue (left, right), one point on the nose tip, and one point on each paw (Figure 5a). For the video of the mouse's body, only the base of the tail is tracked (Figure 5b).

In order to apply the same DLC networks for the left and right side video, their orientation and spatial resolution are matched prior to pose estimation. Concretely, the right video (150 Hz, 640 x 512 px) is flipped and spatially upsampled using ffmpeg to match the left video (60 Hz, 1280 x 1024 px). Since the DLC networks operate on one frame at a time, no additional preprocessing is needed to account for the difference in frame rates.

The computation time for the detection of a point for a given image increases non-linearly with the spatial resolution of the image. For this reason, and the increased ease of creating a hand-labeled training dataset, we devised a method to automatically crop regions of interest for the side view cameras before applying specialized DLC networks to those cropped videos. More specifically, an initial DLC network is trained to find the following anchor points in the side view videos: the top of the drinking spout, nose tip and pupil. These anchor points are then used to anchor rectangular regions of interest, for which the specialized networks are trained: a 100x100 px region around the pupil, a 100x100 px region around the nostril, and a 160x160 px region around the tongue. For paw tracking the whole frame is used, but spatially downsampled to 128x102 px. The original videos are kept at low compression (constant rate factor (crf) 17) on IBL servers for other potential analyses.

We further compute motion energy (pixel average of the gray scale difference of subsequent video frames) for rectangular regions roughly corresponding to the whisker pad and the body of the mouse (indicated in green in Figure 5). Again, these regions are calculated based on anchor points found via the DLC detection.

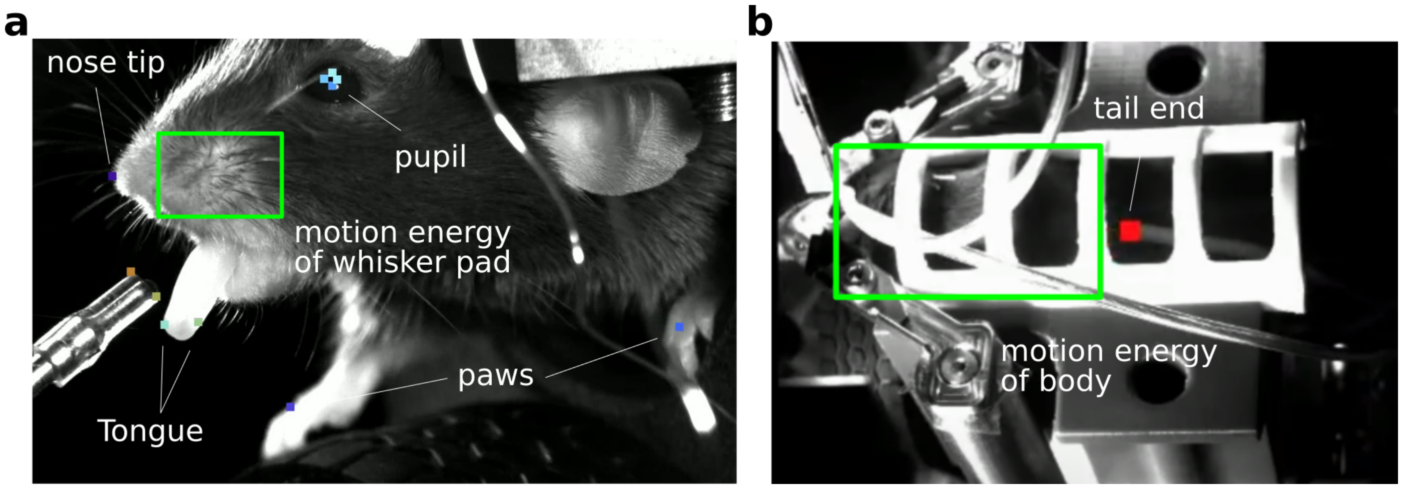


Figure 5. a) Frame of left camera video with location of the points that are routinely tracked for side videos. Green rectangular region indicates the whisker pad region used to compute motion energy. **b)** Frame of body camera video with the location of the tail end which is tracked in this view. Green rectangle indicates the region for which the motion energy of the animal's body is calculated.

Detection of "bad" frames and training set augmentation

DLC network training requires a set of frames that are hand-labeled with the true location of each body part to be tracked. A deep neural network is then trained to transform an input frame into the (x, y) coordinates of each tracked body part in that frame. The IBL dataset consists of videos from different labs and different animals, and despite standardization efforts there is variance across camera placement, brightness, animal appearance and rig details (e.g. drinking spout location and shape, Figure 6). For sufficient tracking accuracy, it is thus necessary to collect a large training set that encompasses this variability. We also found it necessary to implement an iterative process for training set augmentation, which we describe below.

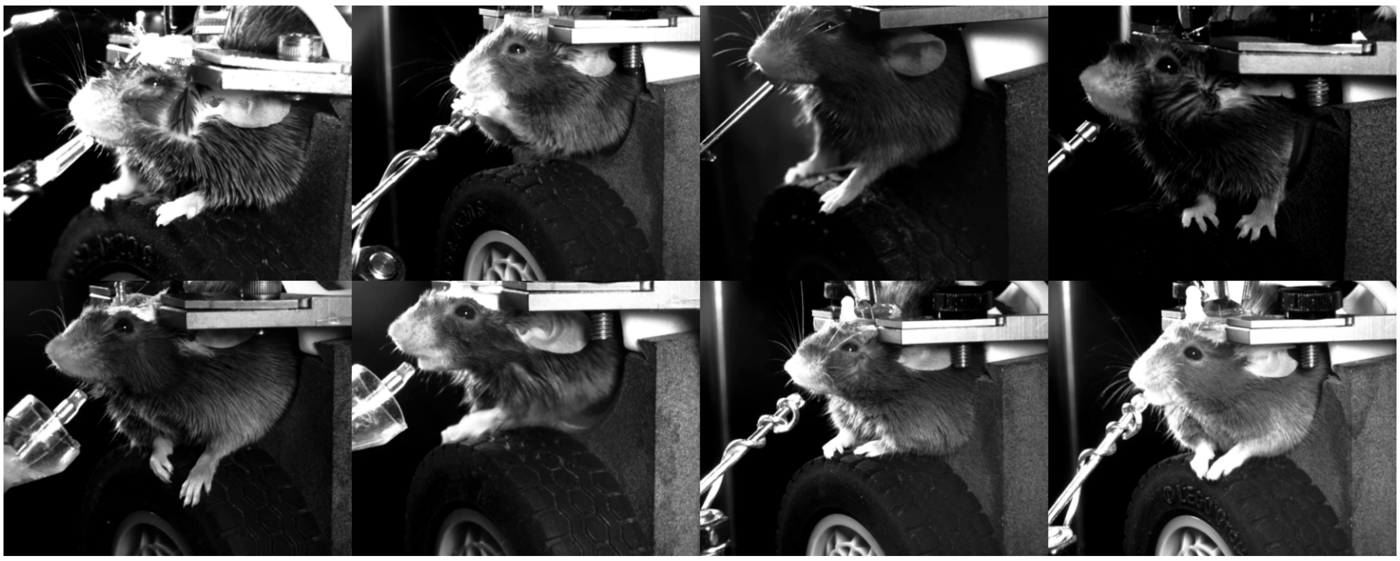


Figure 6. Example frames from the left view camera, illustrating video variance across labs and animals for videos that passed raw video QC.

We began collecting the training frames for each network using the default approach implemented by DeepLabCut: videos from which to draw training frames were selected, then a Principal Component Analysis was performed on a downsampled version of each video, a k-means clustering model was fit in this reduced space, and one frame was selected from each cluster. This approach provides a diversity of poses from each video and serves as a reasonable starting point for training pose estimation networks.

Once a network has been trained, the next challenge is to automatically discover failure cases on new videos, without additional hand labels, and without looking at the predictions frame by frame. As an example, our initial paw network only contained training frames that included one or both forepaws, but we later discovered many sessions where one or both hindpaws came into view, leading to mislabeled body parts. If we can discover failure cases like this, we can label them, add them to the set of training frames, and retrain the model for improved generalization performance. We pursued a variety of approaches for automatically discovering failure cases, described in detail below, and ultimately used several of these approaches across multiple iterations of this training set augmentation process. We often labeled an entire window of frames around the selected failure cases, as this led to a greater diversity of frames and also facilitated the labeling process. This was particularly true for difficult frames where a single time point may not have contained enough information to distinguish, for example, which paw is the left and which is the right.

Below, we describe the frame selection methods for each network. For each selection method, we also require the motion energy of the inferred markers for the selected frames to be above some threshold. This drops redundant “still” frames and again ensures a diversity of frames when we additionally label the frames in a window around the selected frame. For all networks, we randomly sampled frames with low-likelihood predictions (conditioned on high motion energy). We also randomly sampled frames with large jumps in marker position from one frame to the next. In addition to the low-likelihood and jump methods, we employed the following network-specific strategies:

Paw network

- 3D reprojection error: the paws are visible from both left and right cameras (see Figure 3), allowing for the triangulation of the paw position in 3D space. On each frame, the inferred 3D position of each paw can be reprojected back into the 2D pixel space of each camera view, and compared with the 2D output of the pose estimation network. The difference between the original prediction and the reprojection is a useful error metric that can identify new failure modes. In the earlier example of a mislabeled hind paw, the network prediction may be confident (and thus not be picked up by the search through low-likelihood frames) and consistent across multiple frames (and thus not picked up by the jump method). However, if the 2D prediction from the other camera is correct, there will be a discrepancy between the two views, leading to high reprojection error.
- Ensembling error: a classic technique in machine learning is to train an ensemble of models on the same task, with the same data, but introducing randomness through various means, such as using different weight initializations, or serving batches of data to the model in a different order. These different models will lead to slightly different predictions on new data, and we found that we could discover new failure cases by considering frames with high ensemble variability.
- Vertical position discrepancy: we developed this final metric to address a specific type of error that we found across multiple sessions: occasionally when both paws are visible and near the body midline, the network would switch the labels on the two paws. These errors are often not detectable by the standard low-likelihood and jump methods. They can be detected by the 3D reprojection error method, but for videos with many types of reprojection error we found that a more targeted approach is to measure the vertical distance between the left paw and the right paw in both videos, and compare across the two videos. If the paws are mislabeled in one video but not the other, the sign of this value will be opposite across the two videos.

Pupil network

- Circle violations: this metric is specific to our pupil tracking problem, where we track the top, bottom, left and right points of the pupil. We make the assumption that these four points should form a circle, and seek out frames where the ratio of the horizontal and vertical axes is far from unity.

Tongue network

- 3D reprojection error: we use the same approach here as for the paw network.

Nose network: as this network tracks a single, relatively stationary point, we did not need to collect new labels and retrain this network.

Table 3 shows the number of labeled frames for each version of each network as we iterated through the training set augmentation process, along with the number of experimental sessions from which the frames are selected. Figure 7 shows the quantitative improvements of the networks throughout the process.

	Paw		Pupil		Tongue		Nose	
	Frames	Sessions	Frames	Sessions	Frames	Sessions	Frames	Sessions
Iter 0	951	18	924	31	1177	23	754	17
Iter 1	1698	21	2030	45	2323	40	-	-
Iter 2	3798	35	4018	62	3886	58	-	-
Iter 3	7723	60	-	-	-	-	-	-

Table 3. Number of cumulative labeled frames for each iteration of the pose estimation network training set augmentation process, along with the number of experimental sessions from which those frames are drawn.

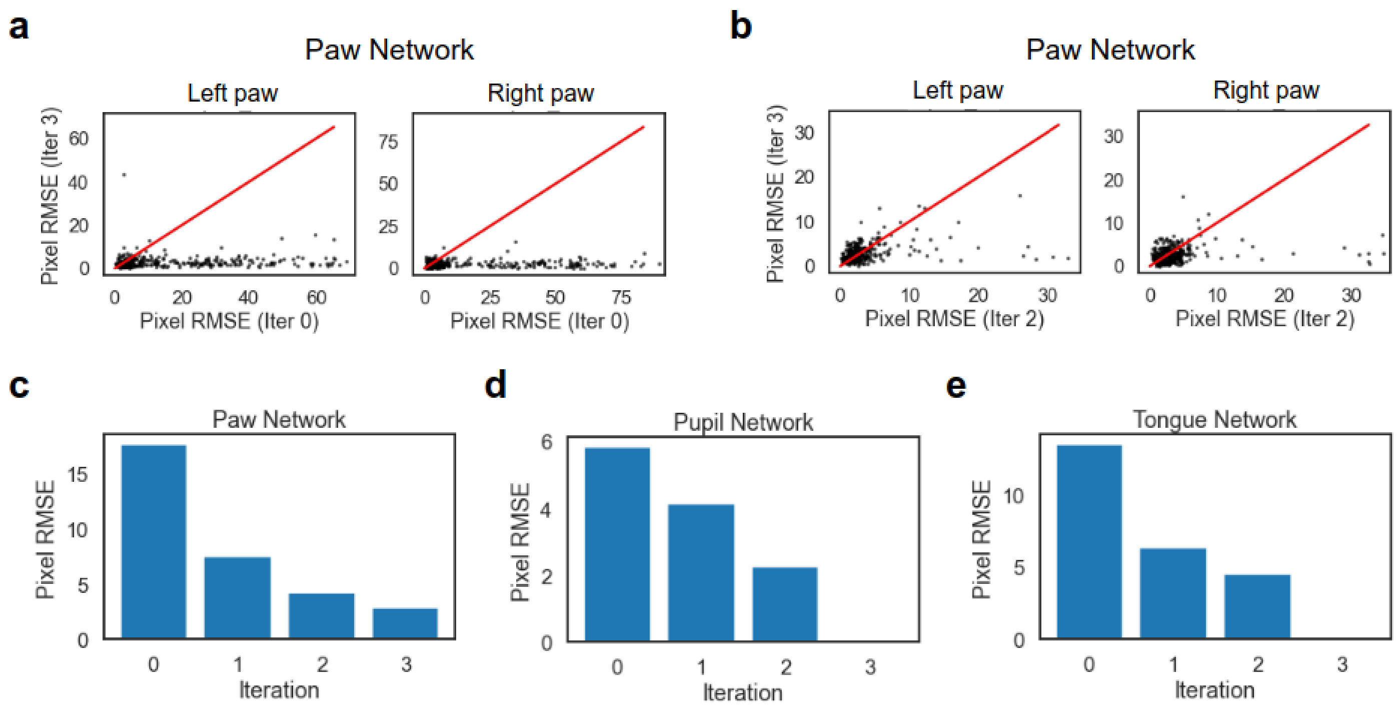


Figure 7. Quantitative improvements of DLC networks throughout the training set augmentation process. **a)** Scatter plot of root mean square error (RMSE) of held-out test frames from the paw network (each dot corresponds to a single frame). Horizontal axis is the RMSE of the initial network (iteration 0), and the vertical axis is the RMSE of the final network (iteration 3). **b)** Same as panel (a), but compares the RMSEs of the final network (iteration 3, vertical axis) and the previous network (iteration 2, horizontal axis). **c)** Mean RMSE across test frames and body parts as a function of training set iteration number. **d, e)** Same as panel (c) but for pupil networks and tongue networks, respectively (note there are only 3 iterations for each of these networks). All RMSE quantities are computed on test frames from the most recent iteration; this will potentially include training frames from earlier iterations, and therefore the decrease in RMSE is a conservative estimate.

Video compression and DLC

Independently and complementary to the recently published mp4 compression test of DLC performance by Mathis et al.[3], we trained DLC to track three facial points (pupil center, tongue, and nose) with h.264 compressed videos at different compression ratios, but identical training frames, using ffmpeg for compression[1].

We found that DLC's performance is not obviously affected even by aggressive compression (mp4, crf=47, Figure 8). However, crf=47 images are blurry and distorted. We therefore consider crf=29 (smaller crf's correspond to higher image quality and lower compression ratios) as the maximum for manually annotating the videos accurately. It is possible to train DLC on weakly compressed videos and then apply it to strongly compressed video. We propose that all recording rig videos are stored at crf=17 for potential future analyses. In case of server storage limitations, videos should be compressed at crf=35, with a small fraction of trials (e.g. every 25th trial) in addition stored as crf=29 for the creation of training sets. This will lead to an overall compression ratio of 100x, corresponding to approximately 330 MB / hour on the recording rig (Table 4).

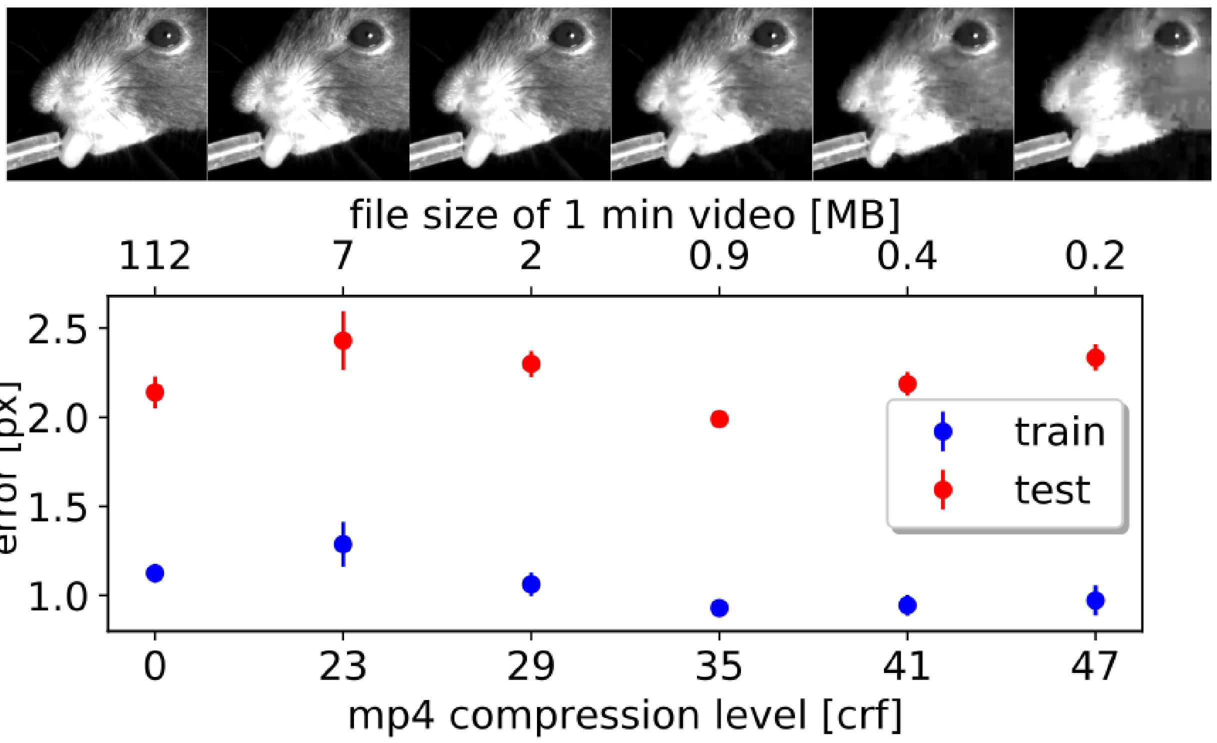


Figure 8. DLC prediction accuracy as a function of h.264 compression level. For each compression level (x-axis) the corresponding file-size is shown (second x-axis) as well as the test and train error of DLC tracking 3 points. The same 200 hand-annotated images were extracted from the video after each compression, using the same position annotations throughout. Then 100 images were used for training a network from scratch, starting with default weights, and tested on the other 100 images. This procedure was repeated 4 times for different random splits of the 200 images into 100 train and 100 test images, each time training a network from scratch with default weights.

[1] e.g. ffmpeg -i input.avi -c:v libx264 -crf 29 output.mp4

(944x956 px)	JPG	crf 29	crf 35
1 h at 150 Hz	18.9 GB	378 MB	189 MB
1 h at 60 Hz	7.6 GB	151 MB	76 MB
Behavior Rig 1 h (1x30 Hz)	3.8 GB	75.5 MB	38 MB
Ephys Rig 1h (2x60 Hz + 1x150 Hz)	34.1 GB	680 MB	341 MB

Table 4. File size for a 1 h side video for different sample rates and mp4 compression levels.

Table 4 lists video file sizes as a function of frame rate and compression level. Note that to a good approximation these file sizes scale linearly with image dimensions and sample rate. Further, increasing crf by 6 reduces the file size roughly by half. Compression via h.264 (mp4) takes changes across frames into account, i.e. areas that do not change much across frames are more strongly averaged over (smeared out) than those where much change happens.

For our current pipeline we compress the videos for the recording rig at crf = 17 in order to keep the option of tracking finer features in the future. The training rig videos are compressed at crf = 29. Table 5 shows the file sizes. Recall that for the training rig there is only one side camera at 30 Hz and 1280 x 1024 px. For the recording rig there are two side cameras (left: 60 Hz, 1280 x 1024 px; right: 150 Hz, 640 x 512 px) and one body camera (30 Hz, 640 x 512 px).

view	length	sampling rate	resolution	crf	file size
side	1 h	150 Hz	640 x 512 px	17	3.3 GB
side	1 h	60 Hz	1280 x 1024 px	17	6.2 GB
side	1 h	30 Hz	1280 x 1024 px	29	0.7 GB
body	1 h	30 Hz	640 x 512 px	17	0.7 GB

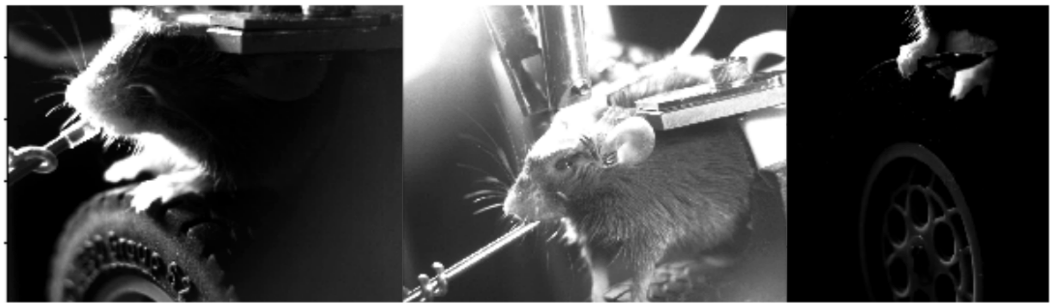
Table 5. File size overview

Quality control

We established automated quality control metrics, some applied to the raw video and some applied to the DLC traces. These metrics serve as an alerting system to flag problematic videos or DLC failure (Figure 9). Details about each metric and the applied thresholds can be found in this document, which also contains links to the relevant code.

In addition to the automatically calculated metrics, we also create an overview plot for each session to facilitate more in-depth visual inspection and can help to uncover issues that are not easily captured in a number. Figure 10 shows such an overview plot for an example session. For each camera an example frame is chosen and the DLC estimates for all tracked points across the entire session are superimposed as scatter plots (Figure 10a-c). These panels help to detect issues with the raw videos as well as the DLC tracking. Other panels show trial averaged behaviors derived from the DLC traces, e.g. licking derived from the tongue and drinking spout points (Figure 10h-i). If these plots vary strongly from the expected patterns, it prompts a more in-depth inspection to determine whether the behavior was indeed aberrant, or if the tracking of the respective points in the videos failed.

a



b

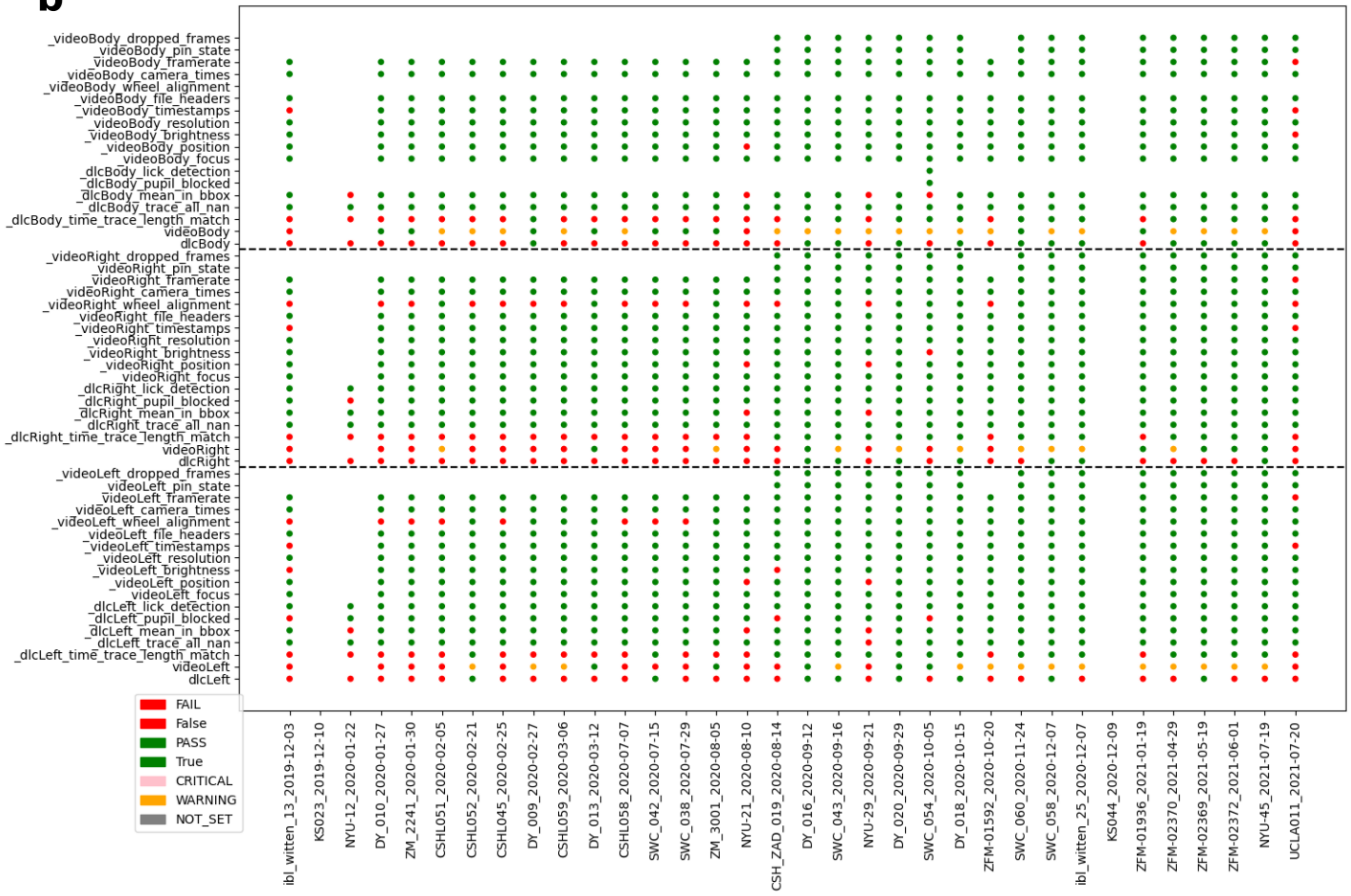


Figure 9. Raw video and DLC quality control criteria. **a)** Example frames of videos that failed the raw video QC criteria as they are too dark (infrared light bulb broke) or misplaced (experimenter accidentally moved the camera). **b)** Temporal evolution of QC criteria for repeated site recordings. A column shows for a given session which of the QC criteria passed (green) and which did not (see legend). The sessions are ordered by recording time, with later sessions having less QC failures as experimenters took feedback into account. The horizontal dotted lines divide the three video types (body, left, right).

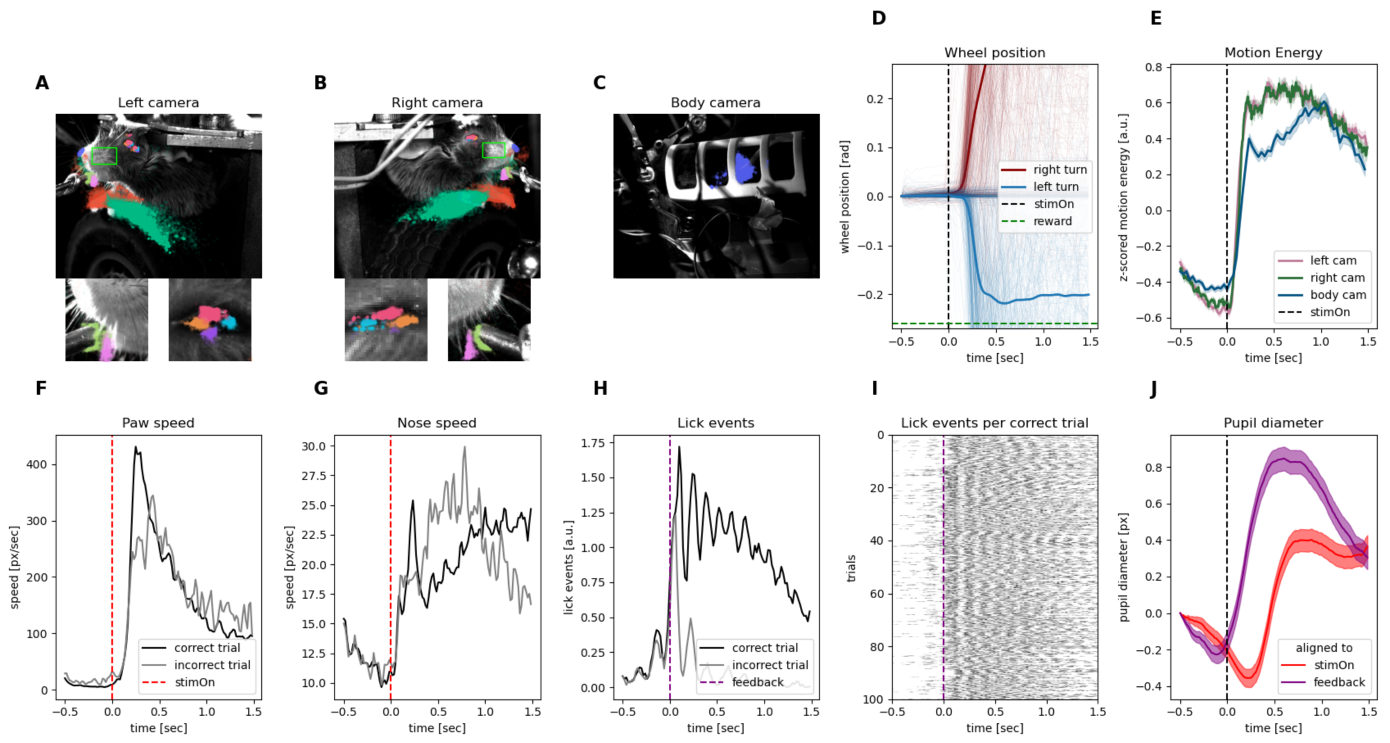


Figure 10. Behavioral overview using DLC traces. **a-c)** An example frame with superimposed DLC traces of the whole session for each camera, additional eye and tongue ROI insets for the side videos. **d)** Wheel position normalized to pre-stimulus baseline, thick lines represent the mean across left/right trials, fine lines are single trials. The colors represent the registered side choice of the animal. **e)** Trial averaged motion energy for the whisker pad area in each side video and for the body in the body video, aligned to stimulus onset. **f)** Right paw speed as a trial average for correct and incorrect trials, aligned to stimulus onset. **g)** Nose tip speed as a trial average for correct and incorrect trials, aligned to stimulus onset. **h)** Trial averaged time binned lick events for correct and incorrect trials, aligned to time of feedback. **i)** Raster plot of time binned lick events for correct trials. **j)** Smoothed estimate of the pupil diameter inferred from up to 4 pupil-delineating points tracked with DLC, aligned to stimulus onset and to feedback time, respectively.

Infrastructure: processing (and reprocessing) many datasets at scale

Within the context of the IBL, the extraction of the DLC markers and quality control metrics has to be performed on hundreds of datasets. Furthermore, the continuous improvement of the DLC networks requires multiple iterations of processing each dataset. We implemented an automated pipeline as part of the IBL data architecture (the architecture is described in detail elsewhere[4]). For the video analysis pipeline in particular, several additional factors needed to be considered. First, the DLC pipeline requires substantial CPU and GPU resources for an extended period of time. For batch (re)processing of the entire set of existing data we therefore chose to employ dedicated compute servers in addition to local servers in each experimental lab. This setup required the implementation of infrastructure aspects that would otherwise be handled by the local IBL processing pipelines, such as data download and job queues. Second, the iteration on the DLC networks made it necessary to implement a versioning system that tracks these network changes, independently of the overall IBL software versioning. Third, the GPU-dependent DLC pipeline required a specific software environment that exceeds the standard IBL dependencies.

We therefore created a dedicated IBL video analysis git repository which contains the code for the core DLC pipeline, version-aware batch processing, integration tests and detailed instructions for installing the necessary environment (https://github.com/int-brain-lab/iblvideo). This repository enables batch processing of existing datasets, e.g. on larger compute servers or in the cloud, as well as continuous processing of newly acquired datasets on the local lab servers.

Current limitations / next steps

The efforts detailed above have produced a robust video analysis pipeline that has been successfully deployed on hundreds of experimental sessions. This pipeline includes pose estimation, motion energy extraction, and automatic computation of quality control metrics and related plots. Nevertheless, opportunities remain for improving and expanding the current pipeline to extract more information from the behavioral videos.

The pose estimation networks are constantly updated in order to improve generalization performance on new animals (Figure 7), which requires reprocessing the entire dataset. This process is slow, requiring several hours per session on a dedicated compute node. Much of the computation time is due to video preprocessing steps with ffmpeg such as flipping, resizing, etc. We are planning to speed up processing by replacing the ffmpeg dependency with an Nvidia DALI data loading pipeline that performs these operations on a GPU rather than a CPU. This will reduce the computational resources needed for reprocessing the entire dataset, saving both time and money.

In addition to improving the speed of the pose estimation networks, there is also room to improve their accuracy. One such avenue is through the use of unsupervised losses on unlabeled video data. This approach allows the network to train on much more data, and can improve pose estimation accuracy if suitable losses are defined[5]. Ensembling is another approach that has the potential to improve accuracy by training multiple models on the same dataset and combining their predictions into a more robust estimate of the pose[6]. Finally, giving the pose estimation models access to multiple consecutive frames provides temporal context that could be useful for resolving errors due to motion blur, occlusions, or other ambiguities present in single frames.

The current video analysis pipeline can be expanded in multiple ways to extract more behavioral information. As a simple example, more body parts such as individual fingers can be tracked, should this information prove useful. There are also many available downstream analyses based on the pose estimates. One such method, the Partitioned Subspace VAE (PS-VAE), extracts unsupervised latent dimensions of the video data that capture variability in the behavior not accounted for by the pose estimates[7]. The resulting combination of supervised pose estimates and unsupervised latent variables provides a richer description of behavior. It is also possible to classify the pose estimates or the PS-VAE outputs into a small set of discrete behavioral classes on a frame-by-frame basis, such as grooming, wheeling, sitting still, etc.

References

1. Mathis, A., Mamidanna, P., Cury, K.M., Abe, T., Murthy, V.N., Mathis, M.W., and Bethge, M. (2018). “DeepLabCut: markerless pose estimation of user-defined body parts with deep learning.” Nature Neuroscience *21*, 1281.
2. Lopes, Gonçalo, et al. "Bonsai: an event-based framework for processing and controlling data streams." Frontiers in neuroinformatics 9 (2015): 7.
3. Mathis, A., and Warren, S. (2018). “On the inference speed and video-compression robustness of DeepLabCut.” bioRxiv 457242
4. Bonacchi, Niccolò, et al. "Data architecture for a large-scale neuroscience collaboration." BioRxiv (2020): 827873.
5. Wu, Anqi, et al. “Deep Graph Pose: a semi-supervised deep graphical model for improved animal pose tracking.“ NeurIPS 2020.
6. Abe, Taiga, et al. "Neuroscience cloud analysis as a service." bioRxiv (2021): 2020-06.
7. Whiteway, Matthew R., et al. “Partitioning variability in animal behavioral videos using semi-supervised variational autoencoders." PLoS computational biology 17.9 (2021): e1009439.