eNeuro

Research Article: Methods/New Tools | Novel Tools and Methods

Real-time closed-loop feedback in behavioral time scales using DeepLabCut

https://doi.org/10.1523/ENEURO.0415-20.2021

Cite as: eNeuro 2021; 10.1523/ENEURO.0415-20.2021

Received: 27 January 2021
Revised: 23 January 2021
Accepted: 26 January 2021

Real-time closed-loop feedback in behavioral time scales using DeepLabCut

Keisuke Sehara[1], Paul Zimmer-Harwood[2], Matthew E. Larkum[1] and Robert N.S. Sachdev[1]

Correspondence to:
Keisuike Sehara (keisuke.sehara@gmail.com) and Robert Sachdev (bs387ster@gmail.com)
Institute of Biology, Humboldt University of Berlin
Charitéplatz 1 (Internal address: Virchowweg 6, CCO Building 03-218)
D-10117 Berlin, Germany

[1]Institute of Biology, Humboldt University of Berlin, D-10117 Berlin, Germany
[2]Department of Physiology, Anatomy and Genetics, University of Oxford, Oxford OX1 3PT, United Kingdom

Abbreviated title: Closed-loop feedback from whiskers in behavioral time scales

Number of pages: 35
Number of figures: 3
Number of tables: 2
Multimedia videos: 2
Number of words for Abstract: 152
Number of words in Introduction: 430
Number of words in Discussion: 1358

Acknowledgements: We thank Johannes Dahmen (Oxford University) for helpful comments on the manuscript, and the Charité Workshop for technical assistance especially Alexander Schill, Jan-Erik Ode and Daniel Deblitz. Marcel Staab and Marti Ritter also contributed to aspects of animal preparation. The authors declare no competing financial interests.

Funding: 1) European Union's Horizon 2020 research and innovation program and Euratom research and training program 20142018 (under grant agreement No. 670118 to MEL). 2) Human Brain Project (EU Grant 720270, HBP SGA1, 'Context-sensitive Multisensory Object Recognition: A Deep Network Model Constrained by Multi-Level, Multi-Species Data' to MEL). 3) Human Brain Project (EU Grant 785907/HBP SGA2 'Context-sensitive Multisensory Object Recognition: A Deep Network Model Constrained by Multi-Level, Multi-Species Data' to MEL). 4) Human Brain Project (EU Grant 945539/HBP SGA3 'Context-sensitive Multisensory Object Recognition: A Deep Network Model Constrained by Multi-Level, Multi-Species Data' to MEL). 5) Deutsche Forschungsgemeinschaft Grant No. 327654276 (SFB1315), Grant Nos. 246731133, 250048060, 267823436, & 387158597 to MEL.

Abstract

Computer vision approaches have made significant inroads into offline tracking of behavior and estimating animal poses. In particular, because of their versatility, deep-learning approaches have been gaining attention in behavioral tracking without any markers. Here we developed an approach using DeepLabCut for real-time estimation of movement. We trained a deep neural network offline with high-speed video data of a mouse whisking, then transferred the trained network to work with the same mouse, whisking in real-time. With this approach, we tracked the tips of three whiskers in an arc and converted positions into a TTL output within behavioral time scales, i.e. 10.5 milliseconds. With this approach it is possible to trigger output based on movement of individual whiskers, or on the distance between adjacent whiskers. Flexible closed-loop systems like the one we have deployed here can complement optogenetic approaches and can be used to directly manipulate the relationship between movement and neural activity.

Significance statement

Here we deploy a deep-neural network-based fast feedback method that can be used to reconfigure feedback to mice based on the movement of particular whiskers, or on the distance between particular whiskers. Our system generates feedback within 10.5 ms. Methods like the one we present here will steadily become part of the standard toolset for manipulating the interaction between the animal and its environment in behavioral time scales.

Introduction

Behavior is a sequence of motor actions controlled and monitored by a pattern of neural activity that is distributed throughout the brain. In most contexts, neural activity related to movement can be detected in a variety of cortical, subcortical, brainstem and spinal circuits. Our toolset for directly manipulating and monitoring neural activity is vast and sophisticated. By comparison, the tools we have for manipulating and monitoring behavior are somewhat meager. Understanding the causal relationship between neural activity and decision making or movement requires tools that work flexibly and rapidly.

High speed videography is a standard tool for monitoring behavior and relating behavior *post hoc* to neural activity. Traditionally, video data have been analyzed manually. More recently, various algorithms have been developed for automating movement detection (Knutsen et al., 2005; Voigts et al., 2008; Perkon et al., 2011; Clack et al., 2012; Ohayon et al., 2013; Giovannucci et al., 2018; Dominiak et al., 2019; Vanzella et al., 2019; Betting et al., 2020; Petersen et al., 2020). With the development of DeepLabCut, a marker-less pose-estimation toolkit based on deep learning (Mathis et al., 2018), computer vision approaches are being used for monitoring poses of animals and for tracking the movement of virtually any part of the body. The main advantage of DeepLabCut over other algorithms is that it can be deployed on any and all parts of the body that are imaged in videos. Furthermore, DeepLabCut is easy to use, and it is easy to train with additional data sets.

These standard approaches for movement tracking have been supplemented or complemented with real-time monitoring and manipulation of behavior (Nashaat et al., 2017; Cao et al., 2018; Sehara et al., 2019). One uses low resolution,

inexpensive, color tracking cameras for monitoring the movement and position of animals at a latency of ~30 ms (Nashaat et al., 2017). A more recent approach used machine learning algorithms that work at a latency of ~ 100 ms or less (Cao et al., 2018). Yet another uses neuromorphic cameras and has a much shorter latency (2 ms) for tracking whiskers (Sehara et al., 2019). Here we describe a real-time DeepLabCut implementation that can be used to track movement at a resolution of 10–15 ms and generate triggers in real time at a latency of 10.5 ms. We performed our proof-of-principle experiments in the rodent whisker system, a complex sensorimotor system which requires tracking of similarly shaped tactile sensors, each moving at 20-25 Hz. We expect our system to be applicable to real-time tracking of movement of any set of body parts in almost any behavioral application.

Materials and Methods

Animal experiments

All procedures using mice were performed in accordance with protocols approved by the Charité–Universitätsmedizin Berlin and the Berlin Landesamt für Gesundheit und Soziales (LaGeSo) for the care and use of laboratory animals..

Animals: Six male C57BL/6 mice (RRID:IMSR_JAX:000664) housed under a 12/12 h reverse day/night cycle were used in this study.

Surgery: Animals were anesthetized with ketamine / xylazine (90 / 10 mg / Kg body weight) and placed on a feedback-regulated heating pad. After subcutaneous lidocaine injection, skin and fascia were removed from the skull. A lightweight aluminum head post was attached to the skull using a Rely-X (3M) cement, followed in some cases by Jet acrylic black cement (Dominiak et al., 2019; Sehara et al., 2019). Animals were monitored during recovery and were given antibiotics (enrofloxacin) and analgesics (Buprenorphine and Carprofen). After a recovery period, the animals were habituated to the experimenter's handling and to being head-fixed at the setup.

Animal imaging setup: The mouse was head-fixed under infrared LED illumination (**Figure 1A**). A 16-bit grayscale camera (DMK 37BUX287, Imaging Source) captured high-speed videos of mouse behavior from the top. For acquisition, a lens with 50 mm focal length was used at f/2.8. The exposure was set at 100–400 microseconds. The acquired frames were transferred via USB3.1 (Gen.1) communication to the host computer. To capture trigger outputs from the real-time tracking program, a 3 mm LED was located on the side of the animal, in the field of view of the camera. Each imaging session lasted for less than 15 minutes.

Imaging system

The host computer (3.6 GHz Intel Core i7-9700K, 64 GB RAM) ran Ubuntu 18.04, and was equipped with the NVIDIA GeForce RTX 2080 Ti graphics card (11GB RAM). We built a Python-based custom program (the "Pose-Trigger" library), along with a thin wrapper python library for DeepLabCut-based real-time tracking (the "dlclib" library) to run image acquisition and feedback generation (**Figure 1AB**). It runs in loops of image acquisition, position estimation, evaluation of positions, and feedback trigger generation, optionally with storage of acquired images and estimated positions into the disk. We designed the program so that each step after image acquisition can be separately turned on and off. To ensure that we achieve the shortest inter-frame intervals, and to keep frames from dropping, software triggers based on the "busy wait" algorithm were generated within the script and were used to capture each frame (**Figure 1C**).

Image acquisition: Acquired frames had dimensions of 640 pixels (width) and 480 pixels (height). We built a custom Cython (RRID:SCR_008466)-based library ("timedcapture") for managing image acquisition on top of the Video4Linux2 (V4L2) layer. Although the library was optimized for the use with our camera hardware, any V4L2-compliant camera can be used. To reduce acquisition latency on the host computer, the frame buffer size was set to one. To detect the timing of frame capture events, the "strobe-trigger" digital output of the camera was turned on during acquisition.

Position estimation: DeepLabCut 2.1.3 (Nath et al., 2019), with CUDA Toolkit 10.1 and Tensorflow 1.13.1, was used to perform marker-less position estimation.

The "batch size" of the deep neural network (DNN) model was set to one, i.e. it was designed to process one frame at a time. On top of the trained DNN model, we added the "GPU-based inference stage" as it was introduced in DeepLabCut 2.1 (Nath et al., 2019). The incoming image was transformed into 8-bit grayscale. Before being fed to the DNN model, the image size was down-sampled to 320 pixels in width (i.e. half the original size), unless otherwise specified, using Python bindings of the OpenCV library (RRID:SCR_015526) (opencv-python, version 3.4.9.33; https://opencv.org/).

Position evaluation: The experimenter could interactively enter the Boolean expression to be evaluated while the program was running (**Figure 1DE**). Thus, theoretically, any evaluation algorithm can be implemented and changed without re-installing or re-configuring of the program itself.

Output trigger generation: The Boolean result of the evaluation was transformed into the TTL high or low level. To achieve this, we used a combination of FastEventServer (https://doi.org/10.5281/zenodo.3843624) and an Arduino-based single-board driver (https://doi.org/10.5281/zenodo.3515999) that enabled trigger generation at a latency of ~ 100 microseconds (Sehara et al., 2019). Briefly, the server program received commands from the client program via the fast User Datagram Protocol (UDP), relayed them to the Arduino-based board via the USB1.1 cable, and finally responded back to the client program in UDP. The output TTL signal was connected to a LED positioned inside the camera's field of view.

Data acquisition steps: During acquisition, the timestamps, frames, estimated whisker positions, and the status of the TTL output were first stored in memory, then once acquisition was finished these data were written to disk as a zip file containing

serialized NumPy arrays (van der Walt et al., 2011). Two types of timestamps were collected: the frame-start timestamp, and the processing-end timestamp. The frame-start timestamp corresponds to the one obtained before triggering acquisition of each video frame. The processing-end timestamp occurred once the frame had been acquired and an output trigger had been generated.

Deep neural network models

We trained DNN models to estimate tips of three arc whiskers with the aid of the DeepLabCut 2.1.3 toolbox (Nath et al., 2019). One model was trained specifically for each mouse. As a base network of the DNN, the default ResNet-50 was used. The labeled whisker tips varied from one animal to another, depending on the length of the animal's whiskers at the beginning of the course of behavioral sessions. We used the B2, C1 and beta, the C2, C1 and beta, or the B1, C1 and beta whiskers to train the DNN model. Each DNN model detected a consistent set of whisker tips throughout the whole period of behavioral recordings.

Training of the models: Before starting any real-time feedback experiments, we ran a behavioral session to acquire videos for offline training of the DNN model. A set of 30–60 second videos at 100 Hz frame rate was acquired from each head-fixed mouse whisking freely without any constraints or task. The mini-batch K-means clustering method was used to extract 60–120 video frames in total from the set of videos to be used as the training data set. Training of the DNN models typically required a total of 1–3 training sessions, each comprising ~ 1,000,000 iterations. After each training session, 30 outlier frames were picked up from each video and

198 added to the training data, using the default “jump” method provided in DeepLabCut
199 2.1 (Nath et al., 2019).
200
201 **Data analysis**
202 Python (RRID:SCR_008394) (version 3.7.7) (van Rossum, 1995) was used to
203 run *post-hoc* analysis. In addition to the Python DeepLabCut toolbox, the following
204 libraries were used during the analysis and annotation: NumPy (RRID:SCR_008633)
205 (version 1.19.1) (van der Walt et al., 2011), SciPy (RRID:SCR_008058) (version 1.5.2),
206 matplotlib (RRID:SCR_008624) (version 3.0.3) (Hunter, 2007), pandas
207 (RRID:SCR_018214) (version 1.0.4) (McKinney, 2010), scikit-image (version 0.17.2)
208 (van der Walt et al., 2014), Neo (RRID:SCR_000634) (version 0.9.0) (Garcia et al.,
209 2014), h5py (version 2.10.0) (Collette, 2013) and Jupyter notebook
210 (RRID:SCR_013984). For *post hoc* annotation of videos, the pillow image processing
211 library (version 7.1.2, https://python-pillow.org/) and the Python bindings of the
212 OpenCV library (RRID:SCR_015526) (opencv-python, version 3.4.9.33;
213 https://opencv.org/) were used.
214 *Latency profiling*: During real-time experiments with the target inter-frame
215 intervals of 10 ms, (a) the “strobe-trigger” digital output from the camera and (b) the
216 pose-trigger output from the Arduino-based board were recorded at 10 kHz per
217 channel using the Power1401 interface (CED, the United Kingdom) and Spike2
218 (RRID:SCR_000903). The resulting Spike2 files (27 sessions in total from 3 animals)
219 were used to compute inter-frame intervals and the pose-trigger latency. For
220 calculation of the time spent for internal procedures of our Python program, we
221 recorded system timestamps using the Python “time” module. To profile the latency

of image acquisition, timestamps were obtained before and after the Python function call to acquire a frame, and their differences were calculated. For estimation of the time spent for position estimation using DeepLabCut, the net duration including OpenCV-based down-sampling and DeepLabCut method calls (i.e. the total time spent to process the frame and obtain estimated positions) were measured. To profile latency for trigger generation, timestamps were collected before and after one transaction of commands, i.e. from the point when the client Python library dispatched the UDP packet to the server, to the point when the server program responded back to the UDP packet after the Arduino-based output board generated the trigger. For experiments when we varied the target inter-frame intervals from 10 ms to 20 ms, the resulting intervals were computed based on the frame-acquisition timestamps of the acquired videos.

Profiling of estimation accuracy: For each animal, 60 additional frames (20 from each 30 s video) were extracted from the video and were manually labeled independently of the training data using ImageJ (RRID:SCR_003070) (https://imagej.nih.gov/). After training of the corresponding DNN model, the frames were subsampled and fed to the model. The resulting whisker positions were compared to the positions of manual labeling to compute error figures. Data from three animals were pooled and summarized together.

Profiling of real-time trigger accuracy: For each animal, we trained a distinct set of DNN models that perform *post-hoc* estimation of the "true" positions of whisker tips in videos of real-time trigger generation. Based on this ground-truth data, kernel-density estimation was performed to generate event-density distributions of the position-values being evaluated in real time (i.e. the position of

the middle whisker tip for the position-based evaluation, and the difference between the positions of the two whisker tips for the spread-based evaluation). A Gaussian distribution with a standard deviation of 0.5 mm was used as the density kernel. To compute the conditional probability of trigger generation at each position, the density distribution during the triggered time points was divided by the distribution of the whole period of acquisition. The cumulative probability distribution of a Gaussian distribution was then fitted to the conditional probability distribution to estimate the position and variability of trigger threshold. The mean of the fitted Gaussian distribution was defined as the detected threshold position (note that closer to the set value is a better estimate), whereas the 2x standard-deviation value was considered to be the variability of the threshold (note that smaller values are more accurate).

To prevent the trigger accuracy figures from varying solely based on position-estimation accuracy, we used videos that had the mean per-frame part-wise estimation error of below 2 mm (whisker position-based triggering, N=15 videos from 6 behavioral sessions from 5 animals; whisker spread-based triggering, N=8 videos from 4 behavioral sessions from 3 animals).

Code and data availability: The code and the software described in our paper are freely available online (Pose-Trigger, https://doi.org/10.5281/zenodo.4459345 ; dlclib, https://doi.org/10.5281/zenodo.4459239 ; timedcapture, https://doi.org/10.5281/zenodo.4459208), and their source packages are available online at Python Package Index (PyPI, https://pypi.org/). The raw video data, the DeepLabCut models and the analytical procedures used in this study are available

freely online (https://gin.g-node.org/larkumlab/RealtimeDLC_DataRepository) and will be available upon request.

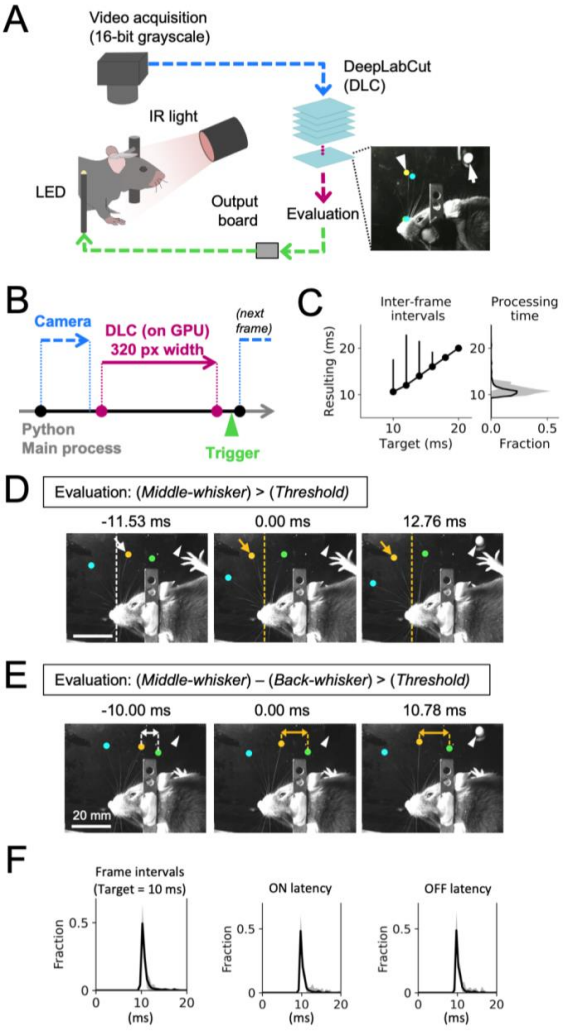


Figure 1. Proof of concept of real-time feedback based on whisker position. (**A**) Schematic of the setup. The mouse was head-fixed under the camera, and high-speed video was acquired under infrared illumination. Whisker positions were estimated from each frame. Digital output — turning on an LED — was generated based on estimated positions using DeepLabCut. The inset shows the example annotations based on the output. Estimated positions were stored after acquisition and used for *post-hoc* annotation. Arrowhead shows the estimated position of a whisker tip. Arrow points to the flashing LED. (**B**) Flow of acquisition and trigger output. Acquisition of a frame (blue) starts with a trigger being generated by the busy-wait algorithm. The acquired frame was passed on to DeepLabCut-mediated body-part estimation (red) after subsampling the frame to half the original size. The status of trigger output was determined based on the estimated body-part positions, and was generated as TTL signal (green). (**C**) Acquisition speed. The effect of changing the busy-wait timer settings (x axis) on the inter-frame acquisition intervals (median and 5% confidence intervals of N=3000–3500 frames per setting). The right panel shows the average histogram for the total processing time per frame (median and best/worst cases out of N=6 sessions). Exposure of 400 µs was used, and frames were subsampled to 320 pixels in width before being processed. (**D, E**) Example consecutive frames in a single representative session. The positions of three whiskers (cyan, orange, green) on one side of the mouse's face were estimated. Flashes of the LED in the field of view (arrowheads) reported the generation of output triggers in real time. Videos in (**D**) and (**E**) differ in the way the estimated whisker positions were evaluated during the session. In (**D**), the trigger (arrowhead) was generated when the middle whisker (orange, arrows) protracted across the arbitrary border (dotted lines). The color of the border and the arrow indicates the status of trigger output (white: off, orange: on). (**E**) The trigger was generated when the horizontal distance (arrows) between the two whiskers on the back (orange, green) went above the arbitrary threshold. The color of the arrow indicates the status of trigger output (white: off, orange: on). Scale bars, 20 mm. (**F**) Latency profile during real-time acquisition and trigger generation. When the inter-frame intervals were targeted at 10 ms the resulting frame intervals (left), the trigger on-event latency (center) and the trigger off-event latency were stable at ~10 ms.

272

Results

Real-time trigger generation based on whisker positions

Mice were head-fixed under the infrared illumination while high-speed video frames were acquired from the camera from above (**Figure 1A**). To monitor the timing of the TTL output, a LED positioned in the field of view of the camera (**Figure 1A**, inset, arrow) was set to flash in response to each trigger, allowing simultaneous capture of whisker positions in the video data and timing of the trigger signal. Our proof-of-concept experiment consisted of three phases: 1) acquisition of training video data for the deep neural network (DNN) model; 2) training of the DNN model using DeepLabCut; and 3) applying the DNN model to real-time body-part estimation. Each training session for the DNN model took ~12 hours.

We set up our acquisition program so that whisker position estimation and trigger output generation were complete before the beginning of the acquisition of the next video frame (**Figure 1B**). With the use of a busy-wait algorithm for triggering the acquisition of the next frame, actual inter-frame intervals varied in the range of 10–20 ms (**Figure 1C, left**, median and 5% confidence intervals are shown for each condition). For the target inter-frame intervals of 10 ms, the resulting intervals were 11.27±2.34 ms (mean±s.d., median 10.62 ms), or 91.02±11.41 Hz. The variability in the inter-frame interval could be reduced by setting target intervals to 18 ms or larger. The resulting intervals then were 18.06±0.79 ms (55.43±1.48 Hz, mean±s.d., median 18.00 ms). These results were consistent with the profile of per-frame total processing time (**Figure 1C, right**) which was 11.34±2.27 ms (mean±s.d., grand

296 average of N=18206 frames collected across sessions with different target interval
297 settings).

298 To demonstrate the real-time flexibility of our program, we trained the DNN
299 model to estimate tips of three whiskers in an arc, and used different evaluation
300 algorithms for generating an output during a single behavioral session (**Figure 1D,E,**
301 **Video 1**). In one configuration, the program was set to detect the position of a
302 particular whisker, so that the LED switched on when the tip of the middle whisker
303 was protracted across an arbitrary border (**Figure 1D**). In another configuration, the
304 distance between the two adjacent caudal whiskers was tracked, and the LED was
305 only activated when the distance between the two whiskers was greater than an
306 arbitrary threshold value (**Figure 1E, Video 2**). In both cases, the LED turned on one
307 frame after the suprathreshold value was detected.

308 To measure the inter-frame intervals and feedback-trigger output latency
309 more directly, we separately recorded the timing of the "strobe-trigger" output from
310 the camera, as well as the timing of the feedback-trigger output at 10 kHz (**Figure 1F,**
311 **Table 1**). Our analyses revealed that, when the inter-frame interval was targeted at
312 10 ms, the actual inter-frame intervals we achieved occurred at ~10.9 ms (91.7 Hz).

(per-session mean±std, ms)	Mean	Std	Min	2.5%	Median	97.5%	Max
Frame intervals	10.90±0.18	1.78±0.37	9.77±0.05	9.91±0.04	10.35±0.10	16.38±1.63	26.95±1.67
On-event latency	10.50±0.26	1.70±0.47	8.38±2.94	9.48±0.05	9.97±0.16	15.44±1.97	20.67±3.41
Off-event latency	10.50±0.29	1.69±0.49	8.35±2.97	9.46±0.61	9.97±0.21	15.03±2.08	21.22±3.27

Table 1. Statistics of latencies. The acquisition interval between frames, on-event latency, and off-event latency were summarized across multiple acquisition runs. For each parameter, the mean (Mean), standard deviations (Std), minimum values (Min), 2.5-percentiles (2.5%), median (Median), 97.5-percentiles (97.5%), and the maximum values (Max) were averaged. N=26 runs from 3 animals (out of 27 runs in total, 1 run was excluded because there were less than 30 ON/OFF events during the acquisition). Means and standard deviations are shown in milliseconds.

313 The feedback-trigger output latency was ~10.5 ms.

314

315

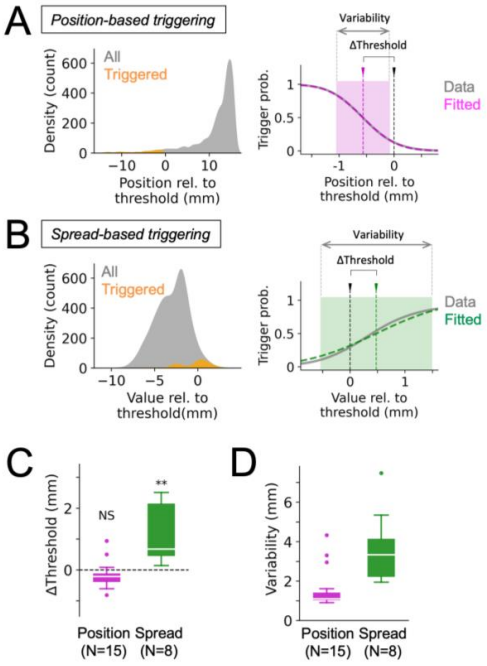


Figure 2. Accuracy of real-time trigger generation. (**A, B**) Estimation of the accuracy of triggers based on video data. Density of occurrence — i.e. the number of frames — was estimated for the whole acquisition period of a video (left panels, gray) and for the period when the trigger was on (left panels, orange). The conditional probability of trigger generation was computed based on the two density distributions (right panels, gray). A Gaussian distribution (right panels, magenta (**A**) or green (**B**)) was fitted to the conditional probability distribution to estimate the difference between the set value and the actual threshold position (ΔThreshold) and the variability in the occurrence of the triggers (Variability, double-headed arrows). The actual and estimated thresholds could vary by 1 mm. The density / counts in (**A**) are from a representative video where position-based triggers were generated, whereas the density / counts in (**B**) are from another representative video where triggers were generated based on the distance between whiskers. (**C, D**) Summary of all position-based and spread-based acquisitions. For both conditions, the difference between the actual threshold position and the set value (**C**) was less than 1 mm on average, even though the detected threshold value was significantly different from the value being set during acquisition (position-based, p=0.0637, NS; spread-based, p=0.0078**; Wilcoxon's signed-rank test). Variability of trigger generation (**D**) was 1–3 mm on average. Compared to the whisker position-based trigger generation, the whisker spread-based triggering was less accurate. N=15 videos (6 behavioral sessions from 5 animals) for position-based trigger generation, and N=8 videos (4 behavioral sessions from 3 animals) for spread-based trigger generation.

316

(values are in mm)		Mean	Std	Min	25%	Median	75%	Max
ΔThreshold	Position-based	-0.18	0.44	-0.81	-0.39	-0.21	-0.11	0.94
	Spread-based	1.10	0.91	0.15	0.47	0.63	1.85	2.44
Variability	Position-based	1.59	1.05	0.90	1.03	1.10	1.41	4.34
	Spread-based	3.69	1.90	1.95	2.21	3.34	4.15	7.50

Table 2. Per-video accuracy of real-time trigger generation. In the top row (ΔThreshold), the average difference between the actual threshold and the value being set during acquisition for each video is shown. The bottom rows (Variability) show the average variability in the threshold in individual videos. N=15 videos (6 behavioral sessions from 5 animals) for position-based trigger generation, and N=8 videos (4 behavioral sessions from 3 animals) for spread-based trigger generation.

317

Accuracy of real-time trigger generation

To examine how accurately triggers were generated in real time, we trained a distinct set of DNN models that perform *post-hoc* estimation of the "true" positions of whisker tips in videos of real-time trigger generation. Using this ground-truth data, event-frequency histograms were generated to estimate the conditional probability of trigger generation at each whisker position (**Figure 2A, B**). Accuracy of real-time trigger generation could be then estimated as the mean and standard deviation of the cumulative Gaussian distribution being fitted to the conditional probability distribution.

For both whisker position-based and whisker spread-based real-time trigger generations, the detected threshold value had a median accuracy of less than ±1 mm (**Figure 2C**; **Table 2**). The position-based trigger generation was less variable (**Figure 2D, left**; **Table 2**) (median 1.10 mm, mean±std 1.59±1.05 mm, N=15 videos) than the whisker spread-based triggering which had a variance of ~3 mm (**Figure 2D, right**; **Table 2**) (median 3.34 mm, mean±std 3.69±1.90 mm, N=8 videos). Both values of variance were well below the total variance in the values being evaluated in real time. The spread-based trigger generation also had higher threshold positions and was more variable compared to the position-based trigger generation. These differences could be related to the fact that the spread-based condition involved the estimation of the positions of two whisker tips, thus doubling the positional error compared to the evaluation based on the position of a single whisker.

Latency profiling of DeepLabCut-based real-time trigger generation

To understand the processes that contribute to the ~10.5 ms latency for generating an output, and to examine whether elements of the system could be sped up, we measured the time spent in each step separately during frame acquisition, whisker position estimation, and trigger generation (**Figure 3**). Acquisition of a video frame, with the size of 640 x 480 pixels, took 2.36±0.02 ms (mean±s.d., median 2.36 ms) for our default configuration of 400 µs exposure. When we lowered the exposure down to 5 µs, it still took 1.97±0.09 ms (mean±s.d., median

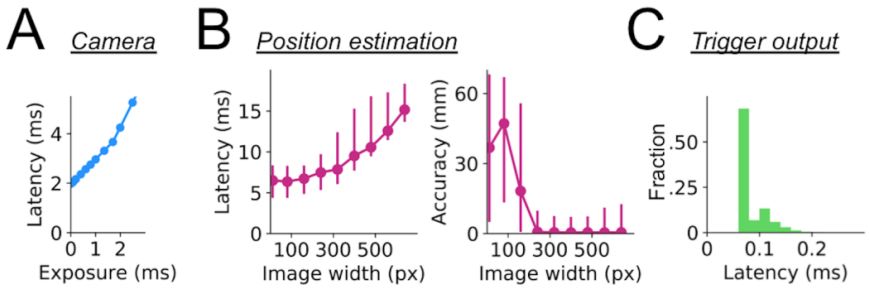


Figure 3. Profiling real-time acquisition procedures. (**A**) The latency to frame acquisition (y axis) for different exposure settings was calculated. Median values from 2000 frames (640 x 480 pixels in size) per exposure setting are plotted. The results were so stable that 5% confidence intervals are not visible in the plot. (**B**) Latency (left) and accuracy (right) of DeepLabCut-mediated position estimation, using our model of whisker-position estimation from 180 frames acquired from 3 animals for each estimation condition. The latency estimation was made with both OpenCV-based subsampling and DeepLabCut-based body-part estimation. Dots represent the median values, and error bars stand for 5% confidence intervals. (**C**) Histogram of latency to output trigger generation. The output was turned on and off repeatedly for 3000 times, and the time spent for the Python call was measured each time.

1.98 ms) to acquire a single frame (**Figure 3A**). The lower bound of ~2 ms presumably corresponds to the time spent in transferring the frame data from the camera to the host computer. Our results are comparable to the theoretical minimum latency of 0.98 ms when a 16-bit image with the size of 640x480 pixels is transferred through USB3.1 communication (which has a theoretical maximum of 5 Gbps for generation 1).

For whisker position estimation, we measured the latency and accuracy at different settings of frame sizes (**Figure 3B**). Here the latency measure included the time spent for frame subsampling and position estimation. We subsampled the frame to the width of 320 pixels during our animal imaging experiments. The latency for this configuration was 8.08±1.44 ms (mean±s.d., median 7.86 ms). Even when we used only 10 pixel-wide frames to pass on to the DNN model, the latency was 6.43±1.03 ms (mean±s.d., median 6.49 ms) (**Figure 3B, left**). On the other hand, the accuracy of the estimation was significantly worse when frame size was reduced to less than half the original size (**Figure 3B, right**).

Output triggers took only 0.10±0.04 ms (mean±s.d., median 0.08 ms, N=3000 frames; **Figure 3C**). This latency was negligible compared with that of the acquisition and estimation steps. Together, under the conditions we used here, we conclude that the current trigger-output latency of 10.5 ms and the current inter-frame interval of 11 ms is close to the limit that our setup can achieve. The results of our profiling imply that there are bottlenecks in the steps of data transfer between different devices.

Discussion

Here we implemented DeepLabCut-mediated real-time feedback generation based on whisker positions. Our system can work at 80-90 Hz and can reliably generate an output in 10.5 ms. These values are within behavioral time scales (Bittner et al., 2017; Isett et al., 2018). This work highlights the fact that real-time feedback generation systems face a trade-off between speed and accuracy. Body-part estimation can be sped up by subsampling video frames, without any large degradation in accuracy. This strategy is likely to be suited to DNN model-based position estimation approaches including DeepLabCut because of their robust estimation based on noisy images.

Challenges towards real-time multi-whisker detection

In the last 20 years, multiple research groups have reported increasingly sophisticated, easy-to-use tools for automated annotation of video data of animal behavior (Knutsen et al., 2005; Voigts et al., 2008; Perkon et al., 2011; Clack et al., 2012; Ohayon et al., 2013; Giovannucci et al., 2018; Dominiak et al., 2019; Vanzella et al., 2019; Betting et al., 2020; Petersen et al., 2020). One natural extension of this ability has been to apply these algorithms for on-line, closed-loop paradigms, where changes in behavior of the animal are detected as rapidly as possible, and the behavior is used to modify or manipulate the brain, the virtual environment or the context of behavior. From this perspective, the rodent whisker system, which has been a model system for understanding brain circuits related to sensory perception, movement and plasticity (Sachdev et al., 2001; Feldman and Brecht, 2005; Brecht et al., 2006; Diamond et al., 2008; Hooks, 2017), poses some unique challenges. Mice

and rats have ~30 similarly shaped whiskers on each side of the face, and they can sweep them back and forth at 10-25 Hz.

Alternatives to video-based real-time tracking do exist to overcome these challenges. For example, it is possible to use EMG for tracking the movement of the whiskers and whisker pad (Kleinfeld et al., 2002; Sachdev et al., 2003). It is also possible to attach a reflective marker to a whisker and optoelectronically track whisker movements reliably at high speeds (Bermejo et al., 1998). But approaches using EMG can be invasive and not targeted to individual whiskers. Tracking with reflective markers can be clumsy and limited to single whiskers. More recently, it has become possible to use neuromorphic approaches to achieve rapid (less than 3 ms) tracking of whiskers (Sehara et al., 2019). Another method has utilized a color-tracking camera to track multiple whiskers (Nashaat et al., 2017), with a latency of ~30 ms, which is 2 to 3 times slower than the ~10 ms behavioral time scale (Bittner et al., 2017; Isett et al., 2018). In addition, this method requires placing UV paint to whiskers every day. Although some of these earlier methods can achieve closed-loop latencies in tracking whiskers that are in behavioral time scales, these methods are either invasive or require markers, or are non-trivial to apply for real time tracking of multiple appendages (i.e. whiskers) moving independently and rapidly.

DeepLabCut and other DNN-based marker-less approaches have a significant advantage over almost all earlier methods for offline pose estimation, and offline body-part tracking. Our current work extends the use of these approaches for real-time marker-less tracking of multiple whiskers. A few recent studies have made forays into real-time tracking with DeepLabCut (Forys et al., 2020; Kane et al., 2020). The first study using this approach demonstrated control of mouse behavior by

419 rewarding particular forelimb movements, achieving a real time tracking latency of
420 50 ms (Forys et al., 2020). The other one, using a different approach, has achieved a
421 latency of 10 milliseconds (Kane et al., 2020). But to achieve a 10 ms latency, Kane
422 and colleagues subsampled each frame and restricted the resolution of video frames
423 to ~ 150 pixels in diameter. While this approach can work for tracking of limbs, it is
424 not likely to be effective for tracking individual whiskers (cf. **Figure 3B**). In addition, it
425 is not clear whether the video frames used for tracking are in fact being saved to disk
426 for additional analysis.

427 One challenge that real-time tracking with video data poses is the
428 computational cost of tracking objects from video frames. Fortunately, the power of
429 processors and algorithms has increased and this has enabled processing of
430 individual frames in less than 10 milliseconds (Knutsen et al., 2005; Voigts et al.,
431 2008; Perkon et al., 2011; Clack et al., 2012; Ohayon et al., 2013; Cao et al., 2018;
432 Dominiak et al., 2019; Betting et al., 2020; Petersen et al., 2020). By comparison,
433 little progress has been made to enhance acquisition latency of frames, i.e. the time
434 required to transfer video frames between devices. As we document here, it can
435 take milliseconds just to obtain a video frame (cf. **Figure 3**). Optimization of
436 hardware / software interaction will be necessary to develop faster real-time
437 tracking algorithms.

438 Here, by optimizing the acquisition and tracking procedures, we have built a
439 system that is capable of closed-loop trigger generation in real time at 10.5
440 millisecond latency. The DNN-based approach affords the possibility of tracking body
441 parts consistently across behavioral sessions without using any artificial markers.
442 Triggers can be generated based on whisker position or on any other features of

443 facial expression that are relevant for inferring the internal states of rodents
444 (Dominiak et al., 2019; Stringer et al., 2019; Dolensek et al., 2020). In addition, our
445 system implements the posture-evaluation mechanism that is easy to edit or update
446 on-line during experiments. The approach we have developed here provides a
447 generic real-time solution for any researcher in the field of behavioral
448 neurophysiology.
449
450 **Bottlenecks of DNN-based tracking approaches**
451 Profiling of our system revealed the major bottlenecks in the DNN-based
452 approaches (**Figure 3**). One major issue is the communication between devices:
453 between the camera and the computer, and between the CPU memory and the GPU
454 memory. The time required for body-part estimation using an image has a lower
455 bound of around 6 ms, regardless of the number of pixels used. Considering the
456 efficiency of the GPU during computation of the DNN model, and the fact that
457 DeepLabCut runs faster in the batch-mode (Mathis and Warren, 2018), it is
458 reasonable to suspect that the transfer of the data to the graphics card is limiting the
459 speed during this process.
460 Similar issues exist in the image acquisition and transfer from the camera to
461 the host computer. Unlike frame-less real-time feedback approaches which can
462 generate trigger outputs in ~2 ms (Sehara et al., 2019), frame-based approaches
463 take milliseconds just to obtain an image from the camera. While most open-loop
464 video applications minimize the data transfer latency by allocating a buffer for
465 incoming video frames, this strategy is not suitable for closed-loop applications that
466 aim to minimize the latency from acquisition of an image to its processing.

There are several additional procedures for reducing the latency to output trigger generation using image-processing based real-time feedback systems. One effective procedure would be the use of more compact DNN models such as MobileNetV2 (Mathis et al., 2019). It could help reduce the load on the GPU and thus could lead to higher efficiency in body-part estimation (Kane et al., 2020). Restricting the region of interest in video frames could also help keep accuracy from degrading while speeding up the estimation process (Nath et al., 2019; Sehara et al., 2019).

It would be also beneficial to use a predictive model to compute future positions of the body parts (Kane et al., 2020). Assuming a certain restricted dynamics in the pattern of movement, predictions could work to detect a stereotypic pattern of movements one frame before it takes place (Kane et al., 2020). But the predictions might not be applicable for fast motion with large degrees of freedom where the implicit assumptions of the model may not always apply.

DNN-based approaches in the field of neuroscience have been evolving rapidly. Based on the same approach we used here on pose estimation, one could well imagine a DNN-based real-time feedback-generation system for other modalities, such as facial expression, vocalization, or population neural activity. Methods like ours are likely to contribute to the interrogation of relationships between virtual worlds, behavior and neural activity.

References

Bermejo R, Houben D, Zeigler HP (1998) Optoelectronic monitoring of individual whisker movements in rats. J Neurosci Methods 83:89–96.

Betting J-HLF, Romano V, Al-Ars Z, Bosman LWJ, Strydis C, De Zeeuw CI (2020) WhiskEras: A New Algorithm for Accurate Whisker Tracking. Front Cell Neurosci 14:588445.

Bittner KC, Milstein AD, Grienberger C, Romani S, Magee JC (2017) Behavioral time scale synaptic plasticity underlies CA1 place fields. Science (80-) 357:1033–1036.

Brecht M, Grinevich V, Jin TE, Margrie T, Osten P (2006) Cellular mechanisms of motor control in the vibrissal system. Pflugers Arch Eur J Physiol 453:269–281.

Cao Z, Hidalgo G, Simon T, Wei S-E, Sheikh Y (2018) OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. arXiv 2017-Janua:1302–1310.

Clack NG, O'Connor DH, Huber D, Petreanu L, Hires A, Peron S, Svoboda K, Myers EW (2012) Automated Tracking of Whiskers in Videos of Head Fixed Rodents. PLoS Comput Biol 8:e1002591.

Collette A (2013) Python and HDF5. O'Reilly.

Diamond ME, von Heimendahl M, Knutsen PM, Kleinfeld D, Ahissar E (2008) "Where" and "what" in the whisker sensorimotor system. Nat Rev Neurosci 9:601–12.

Dolensek N, Gehrlach DA, Klein AS, Gogolla N (2020) Facial expressions of emotion states and their neuronal correlates in mice. Science (80-) 368:89–94.

Dominiak SE, Nashaat MA, Sehara K, Oraby H, Larkum ME, Sachdev RNS (2019) Whisking Asymmetry Signals Motor Preparation and the Behavioral State of Mice. J Neurosci 39:9818–9830.

511 Feldman DE, Brecht M (2005) Map plasticity in somatosensory cortex.
512 Science 310: 810-5.
513 Forys BJ, Xiao D, Gupta P, Murphy TH (2020) Real-time selective markerless tracking
514 of forepaws of head fixed mice using deep neural networks. eNeuro
515 7:ENEURO.0096-20.2020.
516 Garcia S, Guarino D, Jaillet F, Jennings T, Pröpper R, Rautenberg PL, Rodgers CC,
517 Sobolev A, Wachtler T, Yger P, Davison AP (2014) Neo: an object model for
518 handling electrophysiology data in multiple formats. Front Neuroinform 8:10.
519 Giovannucci A, Pnevmatikakis EA, Deverett B, Pereira T, Fondriest J, Brady MJ, Wang
520 SSH, Abbas W, Parés P, Masip D (2018) Automated gesture tracking in head-
521 fixed mice. J Neurosci Methods 300:184–195.
522 Hooks BM (2017) Sensorimotor Convergence in Circuitry of the Motor Cortex.
523 Neurosci 23:251–263.
524 Hunter JD (2007) Matplotlib: A 2D Graphics Environment. Comput Sci Eng 9:90–95.
525 Isett BR, Feasel SH, Lane MA, Feldman DE (2018) Slip-Based Coding of Local Shape
526 and Texture in Mouse S1. Neuron 97:418-433.e5.
527 Kane G, Lopes G, Saunders JL, Mathis A, Mathis MW (2020) Real-time, low-latency
528 closed-loop feedback using markerless posture tracking. bioRxiv
529 2020.08.04.236422.
530 Kleinfeld D, Sachdev RNS, Merchant LM, Jarvis MR, Ebner FF (2002) Adaptive filtering
531 of vibrissa input in motor cortex of rat. Neuron 34:1021–1034.
532 Knutsen PM, Derdikman D, Ahissar E (2005) Tracking whisker and head movements
533 in unrestrained behaving rodents. J Neurophysiol 93:2294–2301.

534 Mathis A, Mamidanna P, Cury KM, Abe T, Murthy VN, Mathis MW, Bethge M (2018)
535 DeepLabCut: markerless pose estimation of user-defined body parts with deep
536 learning. Nat Neurosci 21:1281–1289.
537 Mathis A, Warren R (2018) On the inference speed and video-compression
538 robustness of DeepLabCut. bioRxiv 457242.
539 Mathis A, Yüksekgönül M, Rogers B, Bethge M, Mathis MW (2019) Pretraining boosts
540 out-of-domain robustness for pose estimation.
541 McKinney W (2010) Data Structures for Statistical Computing in Python.
542 Nashaat MA, Oraby H, Peña LB, Dominiak S, Larkum ME, Sachdev RNS (2017) Pixying
543 Behavior: A Versatile Real-Time and Post Hoc Automated Optical Tracking
544 Method for Freely Moving and Head Fixed Animals. eNeuro 4:ENEURO.0245-
545 16.2017.
546 Nath T, Mathis A, Chen AC, Patel A, Bethge M, Mathis MW (2019) Using DeepLabCut
547 for 3D markerless pose estimation across species and behaviors. Nat Protoc
548 14:2152–2176.
549 Ohayon S, Avni O, Taylor AL, Perona P, Roian Egnor SE (2013) Automated multi-day
550 tracking of marked mice for the analysis of social behaviour. J Neurosci
551 Methods 219:10–19.
552 Perkon I, Košir A, Itskov PM, Tasič J, Diamond ME (2011) Unsupervised quantification
553 of whisking and head movement in freely moving rodents. J Neurophysiol
554 105:1950–1962.
555 Petersen RS, Colins Rodriguez A, Evans MH, Campagner D, Loft MSE (2020) A system
556 for tracking whisker kinematics and whisker shape in three dimensions. PLOS
557 Comput Biol 16:e1007402.

558 Sachdev RNS, Berg RW, Champney G, Kleinfeld D, Ebner FF (2003) Unilateral vibrissa
559 contact: Changes in amplitude but not timing of rhythmic whisking. Somatosens
560 Mot Res 20:163–169.
561 Sachdev RNS, Jenkinson E, Zeigler HP, Ebner FF (2001) Sensorimotor plasticity in the
562 rodent vibrissa system In: Mutable Brain (Kaas JH ed), pp152–200. London: CRC
563 Press.
564 Sehara K, Bahr V, Mitchinson B, Pearson MJ, Larkum ME, Sachdev RNS (2019) Fast,
565 flexible closed-loop feedback: Tracking movement in “Real-millisecond-time.”
566 eNeuro 6:ENEURO.0147-19.2019.
567 Stringer C, Pachitariu M, Steinmetz N, Reddy CB, Carandini M, Harris KD (2019)
568 Spontaneous behaviors drive multidimensional, brainwide activity. Science (80-
569) 364.
570 van der Walt S, Colbert SC, Varoquaux G (2011) The NumPy Array: A Structure for
571 Efficient Numerical Computation. Comput Sci Eng 13:22–30.
572 van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N,
573 Gouillart E, Yu T (2014) scikit-image: image processing in Python. PeerJ 2:e453.
574 van Rossum G (1995) Python tutorial. Amsterdam.
575 Vanzella W, Grion N, Bertolini D, Perissinotto A, Gigante M, Zoccolan D (2019) A
576 passive, camera-based head-tracking system for real-time, three-dimensional
577 estimation of head position and orientation in rodents. J Neurophysiol
578 122:2220–2242.
579 Voigts J, Sakmann B, Celikel T (2008) Unsupervised Whisker Tracking in Unrestrained
580 Behaving Animals. J Neurophysiol 100:504–515.
581

582 **Legend**

583 **Figure 1. Proof of concept of real-time feedback based on whisker position.** (**A**)
584 Schematic of the setup. The mouse was head-fixed under the camera, and high-
585 speed video was acquired under infrared illumination. Whisker positions were
586 estimated from each frame. Digital output — turning on an LED — was generated
587 based on estimated positions using DeepLabCut. The inset shows the example
588 annotations based on the output. Estimated positions were stored after acquisition
589 and used for post-hoc annotation. Arrowhead shows the estimated position of a
590 whisker tip. Arrow points to the flashing LED. (**B**) Flow of acquisition and trigger
591 output. Acquisition of a frame (blue) starts with a trigger being generated by the
592 busy-wait algorithm. The acquired frame was passed on to DeepLabCut-mediated
593 body-part estimation (red) after subsampling the frame to half the original size. The
594 status of trigger output was determined based on the estimated body-part positions
595 and was generated as TTL signal (green). (**C**) Acquisition speed. The effect of
596 changing the busy-wait timer settings (x axis) on the inter-frame acquisition intervals
597 (median and 5% confidence intervals of N=3000–3500 frames per setting). The right
598 panel shows the average histogram for the total processing time per frame (median
599 and best/worst cases out of N=6 sessions). Exposure of 400 µs was used, and frames
600 were subsampled to 320 pixels in width before being processed. (**D, E**) Example
601 consecutive frames in a single representative session. The positions of three
602 whiskers (cyan, orange, green) on one side of the mouse's face were estimated.
603 Flashes of the LED in the field of view (arrowheads) reported the generation of
604 output triggers in real time. Videos in (**D**) and (**E**) differ in the way the estimated
605 whisker positions were evaluated during the session. In (**D**), the trigger (arrowhead)

was generated when the middle whisker (orange, arrows) protracted across the arbitrary border (dotted lines). The color of the border and the arrow indicates the status of trigger output (white: off, orange: on). (**E**) The trigger was generated when the horizontal distance (arrows) between the two whiskers on the back (orange, green) went above the arbitrary threshold. The color of the arrow indicates the status of trigger output (white: off, orange: on). Scale bars, 20 mm. (**F**) Latency profile during real-time acquisition and trigger generation. When the inter-frame intervals were targeted at 10 ms the resulting frame intervals (left), the trigger on-event latency (center) and the trigger off-event latency were stable at ~10 ms.

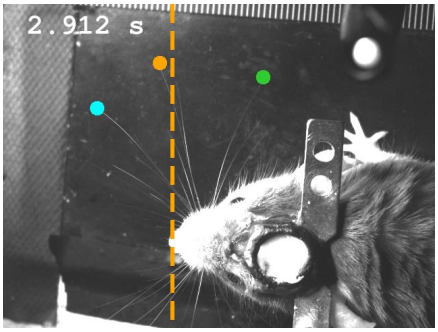
Figure 2. Accuracy of real-time trigger generation. (**A, B**) Estimation of the accuracy of triggers based on video data. Density of occurrence — i.e. the number of frames — was estimated for the whole acquisition period of a video (left panels, gray) and for the period when the trigger was on (left panels, orange). The conditional probability of trigger generation was computed based on the two density distributions (right panels, gray). A Gaussian distribution (right panels, magenta (**A**) or green (**B**)) was fitted to the conditional probability distribution to estimate the difference between the set value and the actual threshold position (ΔThreshold) and the variability in the occurrence of the triggers (Variability, double-headed arrows). The actual and estimated thresholds could vary by 1 mm. The density / counts in (**A**) are from a representative video where position-based triggers were generated, whereas the density / counts in (**B**) are from another representative video where triggers were generated based on the distance between whiskers. (**C, D**) Summary of all position-based and spread-based acquisitions. For both conditions, the difference

between the actual threshold position and the set value (**C**) was less than 1 mm on average, even though the detected threshold value was significantly different from the value being set during acquisition (position-based, p=0.0637, NS; spread-based, p=0.0078**; Wilcoxon's signed-rank test). Variability of trigger generation (**D**) was 1–3 mm on average. Compared to the whisker position-based trigger generation, the whisker spread-based triggering was less accurate. N=15 videos (6 behavioral sessions from 5 animals) for position-based trigger generation, and N=8 videos (4 behavioral sessions from 3 animals) for spread-based trigger generation.

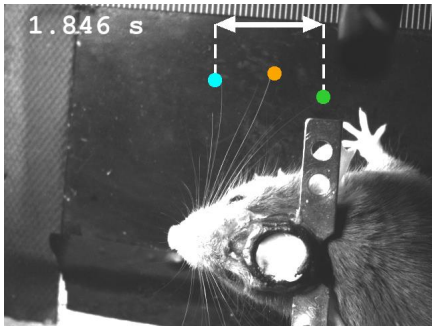
Figure 3. Profiling real-time acquisition procedures. (**A**) The latency to frame acquisition (y axis) for different exposure settings was calculated. Median values from 2000 frames (640 x 480 pixels in size) per exposure setting are plotted. The results were so stable that 5% confidence intervals are not visible in the plot. (**B**) Latency (left) and accuracy (right) of DeepLabCut-mediated position estimation, using our model of whisker-position estimation from 180 frames acquired from 3 animals for each estimation condition. The latency estimation was made with both OpenCV-based subsampling and DeepLabCut-based body-part estimation. Dots represent the median values, and error bars stand for 5% confidence intervals. (**C**) Histogram of latency to output trigger generation. The output was turned on and off repeatedly for 3000 times, and the time spent for the Python call was measured each time.

Table 1. Statistics of latencies. The interval between frames, the on-event latency, and off-event latency across multiple acquisition runs were averaged. For each

654 parameter, the mean (Mean), standard deviations (Std), minimum values (Min), 2.5-
655 percentiles (2.5%), median (Median), 97.5-percentiles (97.5%), and the maximum
656 values (Max) were averaged. N=26 runs from 3 animals (out of 27 runs in total, 1 run
657 was excluded because there were less than 30 ON/OFF events during the
658 acquisition). Values are shown in milliseconds.
659
660 **Table 2. Per-video accuracy of real-time trigger generation.** In the top row
661 (ΔThreshold), the average difference between the actual threshold and the value
662 being set during acquisition for each video is shown. The bottom rows (Variability)
663 show the average variability in the threshold in individual videos. N=15 videos (6
664 behavioral sessions from 5 animals) for position-based trigger generation, and N=8
665 videos (4 behavioral sessions from 3 animals) for spread-based trigger generation.
666

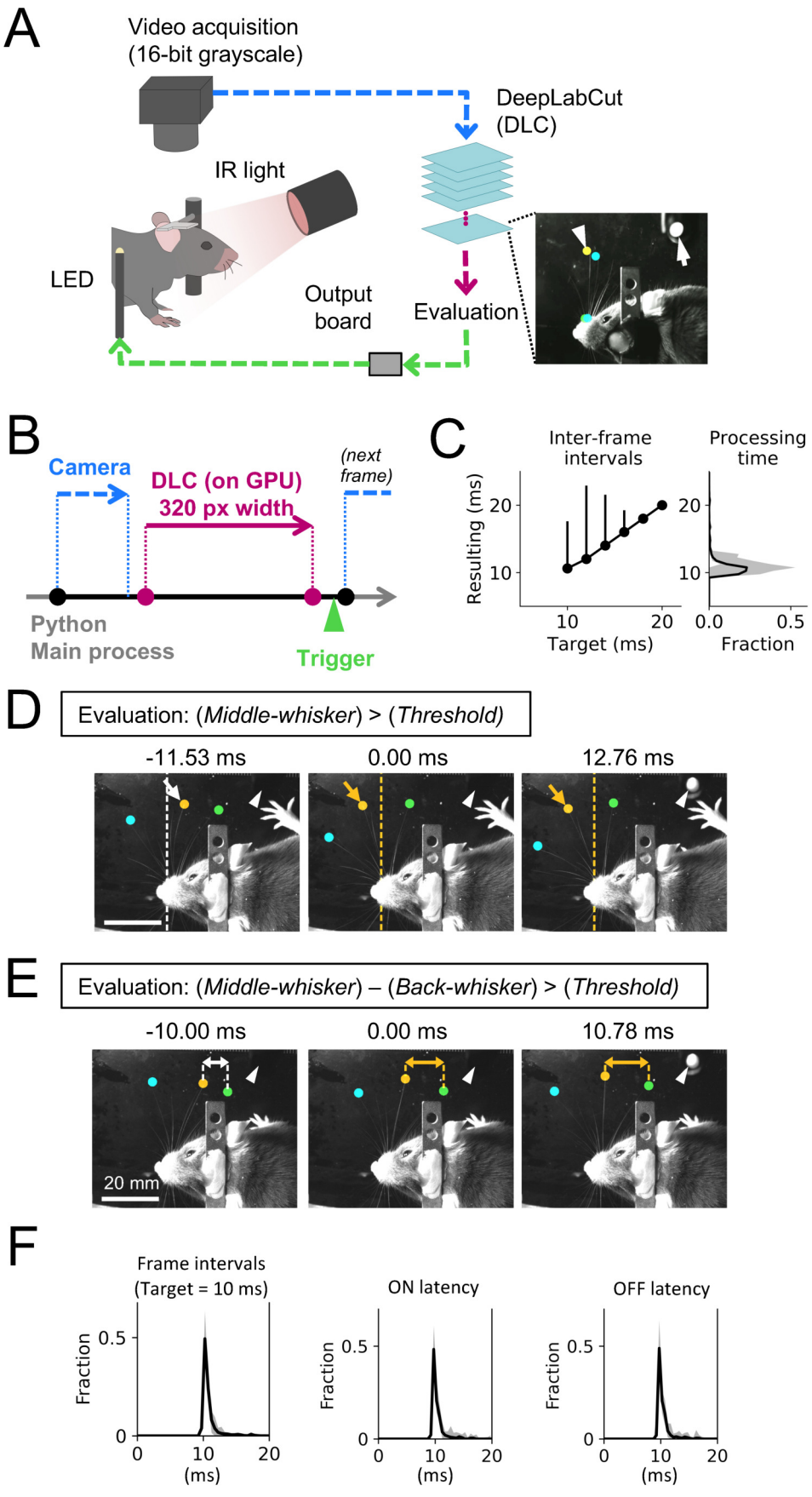


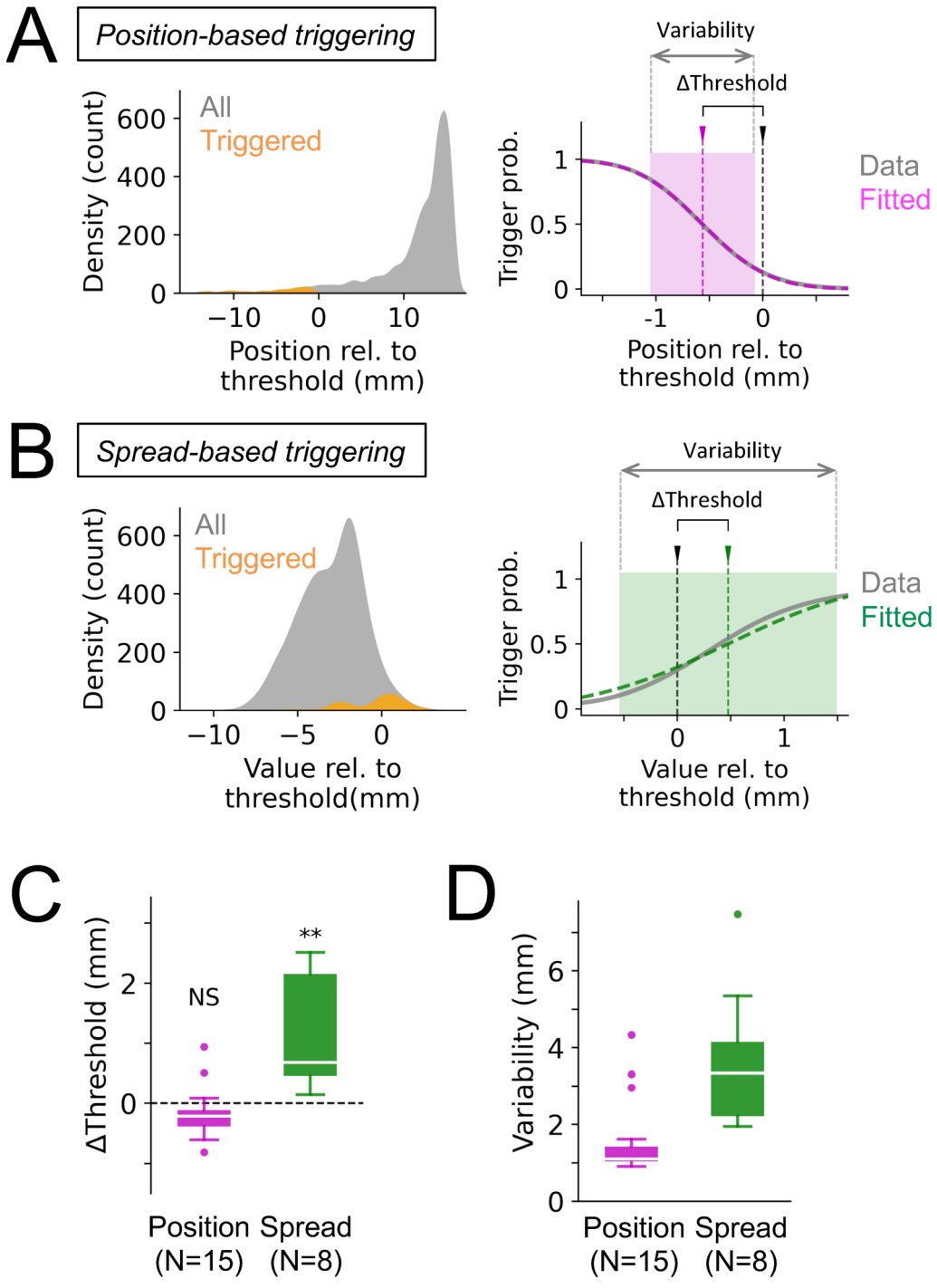
Video 1. Example annotated video in a single representative session, when the position-based evaluation rule was applied. Annotation was performed *post-hoc* based on the acquired data. Dots indicate the positions of three whiskers on one side of the mouse's face. Flashes of the LED in the field of view (top-left) reported the generation of output triggers in real-time when the middle whisker (orange) protracted across the arbitrary border (dotted lines). The color of the border indicates the status of trigger output (white: off, orange: on).

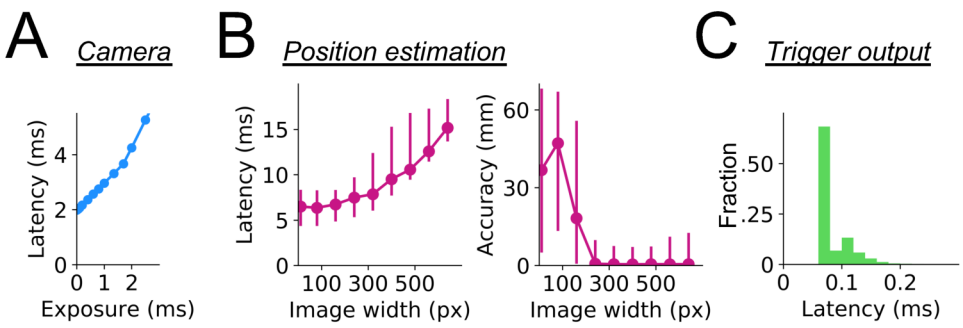


Video 2. Example annotated video in a single representative session, when the distance-based evaluation rule was applied. The video was acquired during the same imaging session as in **Video 1** and was annotated *post hoc*. The trigger was generated when the horizontal distance (arrow) between the two whiskers (cyan,

green) went above the arbitrary threshold. The color of the arrow indicates the status of trigger output (white: off, orange: on). During the time of acquisition, the position estimation of the rostral-most tip (cyan) was unstable. Annotation was turned off when the tip was too caudal, too far behind the caudal-most tip (green).







(per-session mean ± std, ms)	Mean	Std	Min	2.5%	Median	97.5%	Max
Frame intervals	10.90 ± 0.18	1.78 ± 0.37	9.77 ± 0.05	9.91 ± 0.04	10.35 ± 0.10	16.38 ± 1.63	26.95 ± 1.67
On-event latency	10.50 ± 0.26	1.70 ± 0.47	8.38 ± 2.94	9.48 ± 0.05	9.97 ± 0.16	15.44 ± 1.97	20.67 ± 3.41
Off-event latency	10.50 ± 0.29	1.69 ± 0.49	8.35 ± 2.97	9.46 ± 0.61	9.97 ± 0.21	15.03 ± 2.08	21.22 ± 3.27

(values are in mm)		Mean	Std	Min	25%	Median	75%	Max
ΔThreshold	Position-based	−0.18	0.44	−0.81	−0.39	−0.21	−0.11	0.94
	Spread-based	1.10	0.91	0.15	0.47	0.63	1.85	2.44
Variability	Position-based	1.59	1.05	0.90	1.03	1.10	1.41	4.34
	Spread-based	3.69	1.90	1.95	2.21	3.34	4.15	7.50