Ai-based, behavior-dependent approaches for connectomic reconstruction of neuronal circuits

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Vorgelegt von

Jens Florian Schweihoff

aus Essen

Bonn, September 2021

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Heinz Beck [1],

2. Gutachter: Prof. Dr. Ulrich Kubitscheck [2]

[1] Institut für Experimentelle Epileptologie and Kognitionsforschung, Medizinische Fakultät, Rheinische Friedrich-Wilhelms-Universität Bonn.

[2] Biophysikalische Chemie, Mathematisch-Naturwissenschaftliche Fakultät, Rheinische Friedrich-Wilhelms-Universität Bonn

Tag der Promotion: 09.02.2022

Erscheinungsjahr: 2022

"Progress in science depends on new techniques, new discoveries and new ideas, probably in that order"

Sydney Brenner

CONTENTS

FIGURES

TABLES

LIST OF ABREVIATIONS

ABBREVIATION	EXPLANATION
AAV	Adeno-associated virus
ADN	Anterodorsal thalamic nucleus; a brain region containing head direction cells
AI	Artificial Intelligence; field of informatics in which algorithms are developed that display/simulate intelligence and learning
APS	Ammonium persulfate; Chemical compound used in polymerisation reactions
CAG	Strong synthetic promotor used in gene engineering
CAL-LIGHT	Optogenetic system to activity-dependently label neurons after light-induction
CAM	Calmodulin; a calcium-binding protein
CL	Classifier; used in machine learning algorithms to describe classification algorithms
CMOS	Complementary metal–oxide–semiconductor; semiconductor used in image sensors for photodetection
CRE	Cre-Recombinase; Enzyme used in the Cre-loxP expression system
CSV	Comma-separated values; a file format
DAPI	4′,6-diamidino-2-phenylindole; fluorescent dye used to visualize DANN
DAQ	Digital Acquisition Board; Device that converts analogue to digital signals and *vice versa*
DG	Dentate Gyrus; a region in the hippocampus
DLC	DeepLabCut; a software solution for pose estimation of animals
DLC-LIVE	Real-time pose estimation solution of DLC
DLSTREAM	DeepLabStream; a software solution for closed-loop behavior-experiments
EDTA	Ethylenediaminetetraacetic acid; Chemical compound binding iron/calcium that is commonly used in molecular biology
EGFP	Enhanced Green Fluorescent Protein; an enhanced version of GFP
EXM	Expansion Microscopy; a tissue preparation technique (tissue expansion) that is utilized for light microscopy
EXP	Experimental Group
EYFP	Enhanced Yellow Fluorescent Protein; an enhanced version of YFP
FLARE	Optogenetic system to activity-dependently label neurons after light-induction
GPIO	General Purpose Input Output; a digital signal pin on circuit boards
GPU	Graphical Processing Unit
GUI	Graphical User Interface
IP	Internet Protocol address
LEAP	Single animal solution of SLEAP
LOXP	Locus of X-over P1; site used in Cre-loxP expression system
LSFEM	Light Sheet Fluorescence Expansion Microscopy; a combination of LSFM and ExM
LSFM	Light Sheet Fluorescence Microscopy; a microscopy technique

ABBREVIATION	EXPLANATION
MIP	Maximum Intensity Projection; a method for visualization of microscopy data
ML	Machine Learning; see AI
MTURQUOISE2	Blue/turquoise fluorescent protein
NA	Numerical Aperture
OS	Operating System (e.g., Windows)
PBS	Phosphate buffered solution; a common buffer solution in biochemistry
PCR	Polymerase chain reaction; method to quantify/amplify DNA samples
PFA	Paraformaldehyde; Chemical compound used in tissue fixation
RAAV	Recombinant Adeno-associated Virus; a common virus construct used in genome engineering
RAM	Random-access memory; working memory of computers
RGB	Color-space described as values in the red, green, and blue channels
ROI	Region of Interest; descriptive term for region-based analysis
SCFLARE	Optogenetic system to activity-dependently label neurons after light-induction
SDS	Sodium dodecyl sulfate; organic detergent commonly used in molecular biology
SLEAP	Social LEAP Estimates Animal Poses; a software solution for pose estimation of animals
SYN1	Synapsin promotor; commonly used in viral constructs to target neurons
TDTOMATO	Red fluorescent protein
TEMED	Tetramethylethylenediamine; Chemical compound used in polymerisation reactions
TEMPO	(2,2,6,6-Tetramethylpiperidin-1-yl)oxyl; Chemical compound used in polymerisation reactions
TEV	Tobacco etch virus
TRE	Tetracycline Response Element; a component of the Tet-On/Off expression system
TTA	Tetracycline transactivator; a component of the Tet-On/Off expression system
TTL	Transistor-transistor logic; form of digital signal (0 or 1)
USB	Universal Serial Bus; industry standard for cables/connectors
WPRE	Woodchuck Hepatitis Virus Posttranscriptional Regulatory Element; a DNA sequence that is used in viral constructs to enhance expression levels
XFP	X Fluorescent Protein; X as a placeholder for a color e.g., Green

1 Abstract

Characterizing the functional architecture of neuronal circuits that underly complex behavior requires identifying active neuronal ensembles during behavioral expressions of interest. The recent development of light-induced, activity-dependent labeling enables to capture active neuronal ensembles dependent on ongoing behavior, effectively allowing the behavior-dependent, causal identification of relevant structures for subsequent investigation.

However, the behavior-dependent labeling of active neuronal ensembles was limited so far by a lack of dynamic closed-loop feedback systems that reliably detect unconstrained behavioral expressions. To solve this, I developed DeepLabStream (DLStream). DLStream is a versatile closed-loop toolkit providing real-time pose estimation of animals and conducting behavior-dependent experiments. DLStream has a temporal resolution in the millisecond range, is published open-source, and integrates other open-source projects such as deep learning-based pose estimation networks (DLC, SLEAP, DeepPoseKit), GPIO control (Arduino, Raspberry Pi), and machine learning-based behavior classification (B-SoiD, SimBA). To demonstrate DLStream's capabilities, I used the toolkit to label neuronal ensembles active during specific head directions utilizing Cal-Light, a light-induced, activity-dependent biomolecular labeling system. Behavior-dependent light stimulation resulted in labeling of neuronal ensembles active during specific episodes of head direction. Importantly, this experimental strategy has the potential to untangle previously unknown causal relationships. This can be achieved by combining connectomic analysis of the captured ensembles and consecutive manipulation of their neuronal activity.

Additionally, I established the Tetbow system, a virus-mediated, multicolor labeling system that can eventually be combined with behavior-dependent labeling to allow the anatomic analysis of large-scale tissue samples with behavior-dependent, uniquely labeled neuronal ensembles. Here, the focus lay in the effective use of Tetbow labeled samples in a collaborative attempt to develop an automatic segmentation tool to segment uniquely colored neurons in large tissue samples. Notably, some of the results of this thesis were published, including additional experiments using DLStream [1].

2 Introduction

A fundamental goal in neuroscience is to explain how structured neuronal activity gives rise to behavior [2–6]. The initial approach is often to investigate how behavior manipulation affects neural activity. However, techniques that directly manipulate neuronal activity enable modulating the source of behavior and investigate their causal relationship [7]. For this, functional neuronal circuits are often represented as mechanistic models in which components interact in a causal, often linear way. This mechanistic perspective allows probing presumed functions by manipulating components and measure their effect on the overarching network, including their behavioral output [8]. Therefore, by probing how neuronal activity patterns contribute to behavior, mechanistic models of the causal relationship of behavior and neuronal activity can be generated and used to explain the roles of distinct circuit elements [8].

However, the active neuronal ensembles, or functional ensembles, need to be identified and selectively targeted for measurement and manipulation. Unfortunately, the search for these functional ensembles is currently limited by the typically inferior temporal precision of methods dissecting behavior. Optimally, the respective behavioral expressions and corresponding neuronal ensembles should be characterized with a temporal resolution that allows probing the causal links during ongoing behavior [5–7].

In this line, available labeling and manipulation of functionally active ensembles are currently limited by the lack of dynamic solutions that allow behavior-dependent feedback. Thus, two main requirements arise to identify active neurons during specific behavioral expressions and label them for future selective manipulation, imaging, and connectomic analysis.

2.1 Labeling of functionally active ensembles

The first requirement concerns the method of selectively labeling active neurons.

Classic manipulations of larger-scale neuronal activity such as lesions, transgenic alterations, and pharmacological injections cannot identify neuronal ensembles selectively. Additionally, they result in long-lasting and sometimes chronic changes in the investigated animals, making it challenging to interpret behavioral effects and potential side effects on local network structures [7–9]. In contrast, optogenetic manipulation [10–12] offers high temporal precision for fast, short-lived manipulation of neuronal activity [7,8] and has been applied in several fields, such as investigating mechanisms of learning and memory [13–15], perception[16,17], motor control [18,19], and epilepsy [20–23]. Such techniques offer a temporal resolution precise enough that the triggered effect can match the timescale of either behavioral expression or neuronal computation [24,25]. For the analysis of functional ensembles, recently developed optogenetic tools enable the labeling of active neuronal ensembles during episodes of behavior [26–29]. Cal-Light [26,27], for example, allows to virtually time-lock activity connected to behavioral expressions by utilizing a light-induced, activity-dependent expression of reporters (see chapter 3.1).

However, while the system has the molecular contrast and coincidence detection necessary to identify active ensembles during ongoing behavior, the effective use of Cal-Light is currently limited by the lack of dynamic closed-loop feedback systems that detect unconstrained behavior.

2.2 Detection of behavioral expressions in real-time

Therefore, the second requirement to label active neurons during specific behavior is the reliable detection of relevant behavioral episodes in real-time. Preexisting systems that allow behavior-dependent feedback often rely on specialized, on-purpose setups, including intricate beam brake designs, treadmills, levers, and virtual reality setups to approximate the movement of the investigated animal in a given environment and then react accordingly [30–38]. However, the identification of truly unconstrained behavior would facilitate a combination of dynamic behavior-dependent light stimulation and activity-dependent labeling techniques to study neuronal ensembles active during selected behavioral expressions in a previously unmatched level of detail and range.

Fortunately, recent developments in neuroethology have made pose estimation of several species possible using robust deep-learning-based markerless tracking [39–46]. DeepLabCut (DLC) [39,45–48], for example, uses trained deep neural networks to track the position of user-defined body parts and provides motion tracking of freely moving animals (see chapter 3.2). Most interesting, *post hoc* analysis using deep learning-based pose estimation was recently shown to outperform previous go-to commercial solutions [49]. Additionally, sophisticated machine learning approaches have allowed for disentangling the complex behavioral expressions of animals into patterns of reoccurring modules [50–56] (see chapter 3.3). Many of these techniques involve the initial estimation of pose information utilizing toolkits like DLC. However, the leap to behavior-dependent closed-loop experiments using online pose estimation has begun only recently.

2.3 Imaging functional ensembles

Finally, the connectomic analysis of active ensembles, labeled by a combination of Cal-Light and a behavior-dependent closed-loop solution, would benefit from imaging critical elements from mesoscopic (large scale networks) to nanoscopic scale (synaptic level). Thus, allowing a complete characterization of functional ensembles, including the individual connectivity and morphology of neurons. A recently developed, virtual super-resolution imaging technique enables further insight into anatomical details on a small to larger scale at a feasible speed [57,58]. Light sheet Fluorescence Expansion Microscopy (LSFEM, see chapter 3.4) allows studying partial synaptomes with simultaneous ability to zoom out and look at the functional projectome of large-scale networks.

Here, the available resolution of LSFEM will be beneficial in the investigation of large-scale effects by small-scale morphology changes. For example, in brain disorders associated with abnormal dendritic spines [59–61]. However, the efficient identification and tracing of multiple individual neurons within a population is limited by the ability to distinguish between closely neighboring cells. This limitation is severe in regions where the densely layered neuronal architecture results in bundled axonal projections or heavily entangled dendritic trees such as the hippocampus.

Countering these challenges, a biomolecular technique called Tetbow uses the stochastically distributed expression of multiple, differently colored fluorescent proteins [62–65] (see chapter 3.5). However, its effective use with expanded tissue has not been shown yet.

2.4 Goals

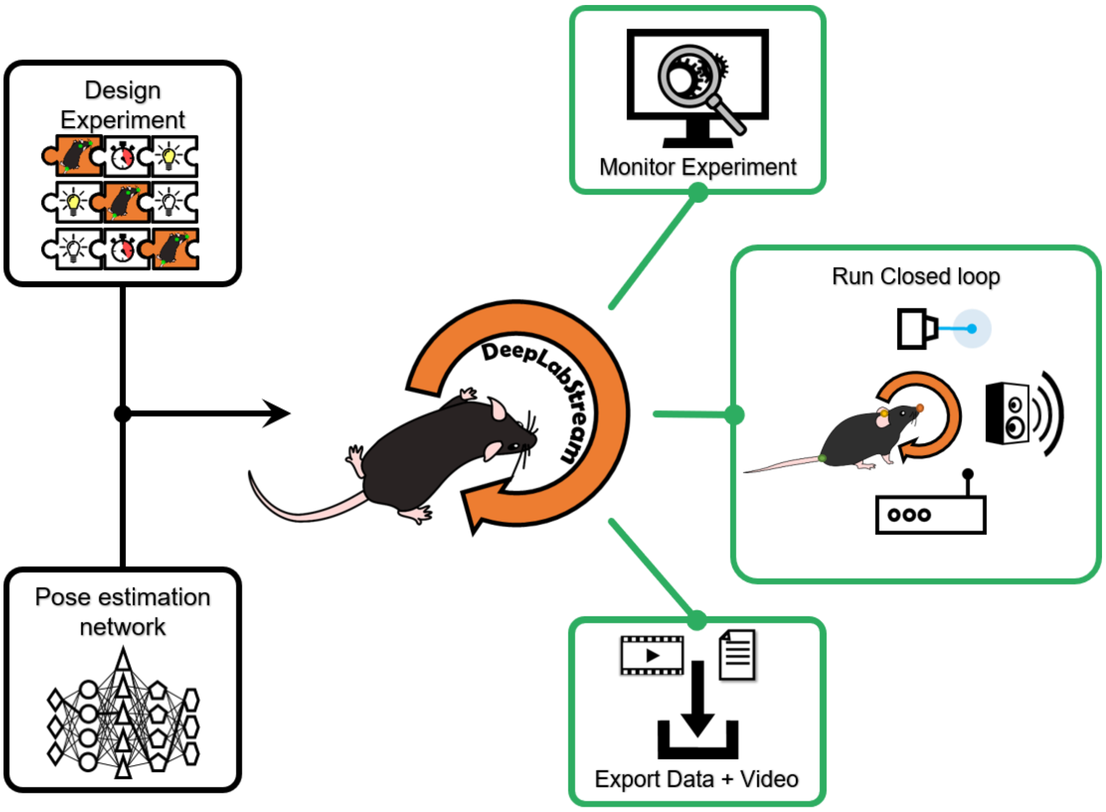


Figure 1 - A visual representation of DLStream.

Visual representation of workflow in DLStream. Initially, an experimental protocol is designed using a sequence of modules (puzzle pieces), and a trained pose estimation network is integrated into DLStream. Afterward, DLStream provides three different outputs for every experiment. 1. Experiments can be monitored on a live stream. 2. The experimental protocol is run based on posture detection. 3. Recorded video and experimental data are exported after the experiment is done.
A version of this figure was also published in Schweihoff, et al. 2021.

Investigating causal links between behavioral expressions and their active neuronal correlates in the brain requires novel techniques with high temporal resolution [2–6]. The development of behavior-dependent circuit labeling will allow novel insights into structure/function relationships within the rodent brain. It promises to bridge connectomics and physiology with the potential to reveal how functional architectures control neuronal computations and behavioral output.

In this thesis, an AI-based, real-time closed-loop system was developed to further investigate neuronal networks correlated to behavioral episodes of interest and used to label neuronal ensembles that were active during ongoing, selected behavior. The software developed during this thesis was designed and published as an open-source toolkit to facilitate a long-term, sustainable software solution [1,66]. Thus, it will continue to benefit from community-driven improvements and extensions.

DeepLabStream (DLStream, Figure 1) is a multi-purpose software solution that enables markerless, real-time tracking and behavior-dependent manipulation of freely moving animals during ongoing experiments. Its core capability is the orchestration of closed-loop experimental protocols, incorporating real-time feedback to facilitate dynamic experimental paradigms. DLStream utilizes state-of-the-art pose estimation such as DLC [39,46–48] to track the postures of mice in real-time and supervises behavior-dependent feedback to input and output devices. It can be combined with biomolecular tools such as Cal-Light to map active neuronal circuits selectively. DLStream's capabilities are demonstrated in a head direction-dependent optogenetic stimulation experiment labeling neurons active during specific head direction. To further establish DLStream as a sustainable software solution, this thesis will elaborate on the versatility of DLStream to adapt to different experimental conditions and hardware configurations and introduce the design of DLStream controlled experiments and triggers. An extensive guide on using and customizing DLStream with several examples is also published alongside the software repository [66].

Additionally, this thesis will give an outlook on the ongoing developments that combine the powerful imaging technique LSFEM with the high-contrast biomolecular tool Tetbow in a collaborative effort to advance the automatic segmentation of large-scale tissue. For this, Tetbow-based multicolor labeling was optimized for use with tissue expansion protocols to lay the foundation for advanced connectomic analysis, eventually combining automatic segmentation of large-scale tissue with behavior-dependent, multicolor activity labeling.

3 Theoretical background

3.1 Translating neural activity into gene expression

In vivo single-unit recording [67], along with recent advances in *in vivo* voltage imaging [68] and miniaturized calcium imaging techniques [69–71], facilitate real-time measurements of neuronal activity in freely moving mice. These techniques provide a

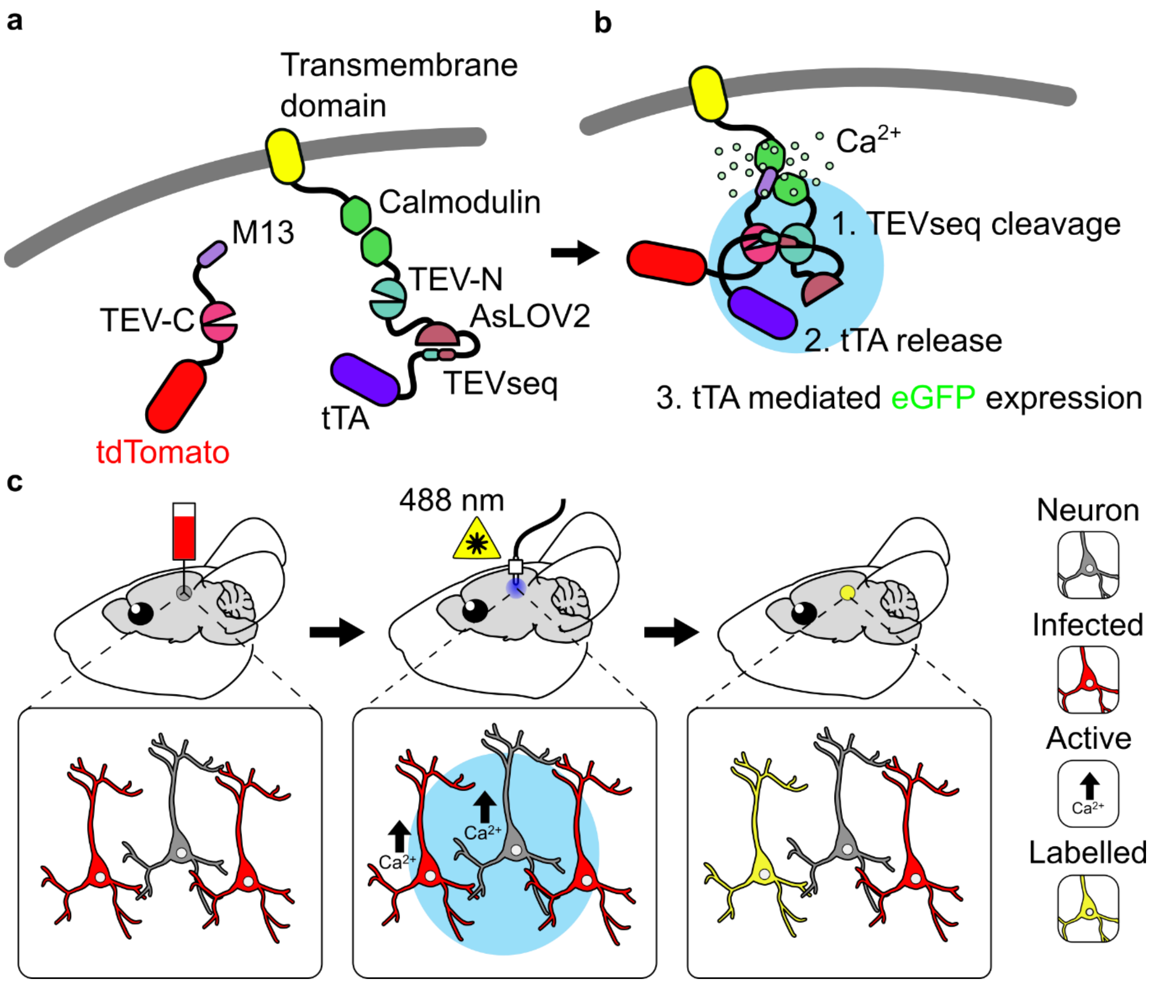


Figure 2 - Schematic representation of Cal-Light

a-b, Schematic representation of the biomolecular mechanism of Cal-Light. Upon Ca^{2+}-dependent binding of Calmodulin and M13, the split TEVp units TEV-C and TEV-N regain function. Simultaneous stimulation by blue light mediates the release of the TEVseq site and enables the release of tTA upon cleavage by TEVp. tTA-dependent eGFP expression is then initiated, labeling the cell green (eGFP) in addition to red (tdTomato).
c, Schematic representation of the experimental progression with Cal-Light. First, mice are injected with a viral mixture (AAV-TRE-EGFP, AAV-M13-TEV-C-P2A-TdTomato, AAV-TM-CaM-NES-TEV-N-AsLOV2-TEVseq-tTA). After infection, neurons are labeled with tdTomato (left panel; red cells). During blue light stimulation (middle panel; blue circle), active neurons (Ca^{2+}, black arrow up) are labeled with eGFP (green), resulting in yellow labeled neurons (right panel), effectively labeling neuronal ensembles active during light stimulation.

wide-ranging foundation for the correlation of recorded neuronal activity and complex behavior. With the development of activity-dependent labeling techniques [26–28], the integration of behavior-dependent circuit labeling is imminent.

The activity-dependent, light-induced labeling technique Cal-Light allows to label neurons active during episodes of behaviors of interest [26]. Cal-Light utilizes a combination of specialized proteins to obtain its coincidence detection (see Figure 2). Its core function is the light-induced translation of cytosolic Ca^{2+} events into gene expression. To accomplish this, a tetracycline-controlled transcriptional activator (tTA) is tethered to the outer cellular membrane and fused with a tobacco etch virus protease (TEVp) cleavage sequence (TEVseq). Cal-Light's light sensitivity is achieved by light-induced TEVseq cleavage and resulting tTA release. By masking the sequence within the C terminus of the Jα-helix of *Avena sativa* phototropin 1 light-oxygen-voltage 2 domains (AsLOV2) 16, the cleavage site is only available for TEVp activity after a blue light-induced conformation change. Activity dependency is realized by splitting the corresponding protease TEVp into N- and C-terminal fragments (TEV-N and TEV-C) that regain proteolytic function upon binding of a Ca^{2+} sensor pair (CaM and M13), which bind upon cytosolic Ca^{2+} rise (see Figure 2 a, b). In its basic configuration, Cal-Light can be combined with tTA-dependent vectors such as rAAV-TRE-eGFP to facilitate the expression of reporter genes (e.g., eGFP). In a behavior experiment, mice injected with Cal-Light can be stimulated with blue light through an implanted light fiber, depending on their behavior (e.g., pulling a lever). Infected neurons in the light-stimulated brain region will express the reporter protein (eGFP) if active during the detected behavioral episode. Infected but inactive neurons remain only labeled with tdTomato (see Figure 2 c). For high molecular contrast, the Cal-Light system needs to be activated repetitively. While this lowers the probability to label behavioral episodes with a low frequency of occurrence effectively, it also increases the contrast between truly correlated and sporadically active neurons. Neurons that are randomly active during the behavioral episode are likely filtered out by the repetitive activation threshold. Cal-Light thereby effectively enables the labeling of active neuronal ensembles during behavioral episodes of interest.

However, the system can also be used to express optogenetic tools to enable behavior-dependent manipulation of neuronal activity [26]. For example, a combination of Tetbow (see chapter 3.5) and Cal-Light would allow to behavior-dependently capture neuronal ensembles with simultaneous multicolor labeling of individual neurons for advanced segmentation. For a more detailed description of the expression system (Tet-O), please refer to chapter 3.5.

As previously stated, for the effective, transient expression of reporter genes, Cal-Light must be reliably activated. Neuronal activity and light stimulation need to coincide, repetitively, with high temporal precision [26]. This requirement renders the technique dependent on external trigger systems to detect behavioral expressions and give instantaneous feedback. In other words, while Cal-Light's ability to detect neuronal activity is remarkable, a major limitation for the effective application of the technique is identifying reoccurring episodes and reliably trigger light stimulation.

3.2 Markerless pose estimation

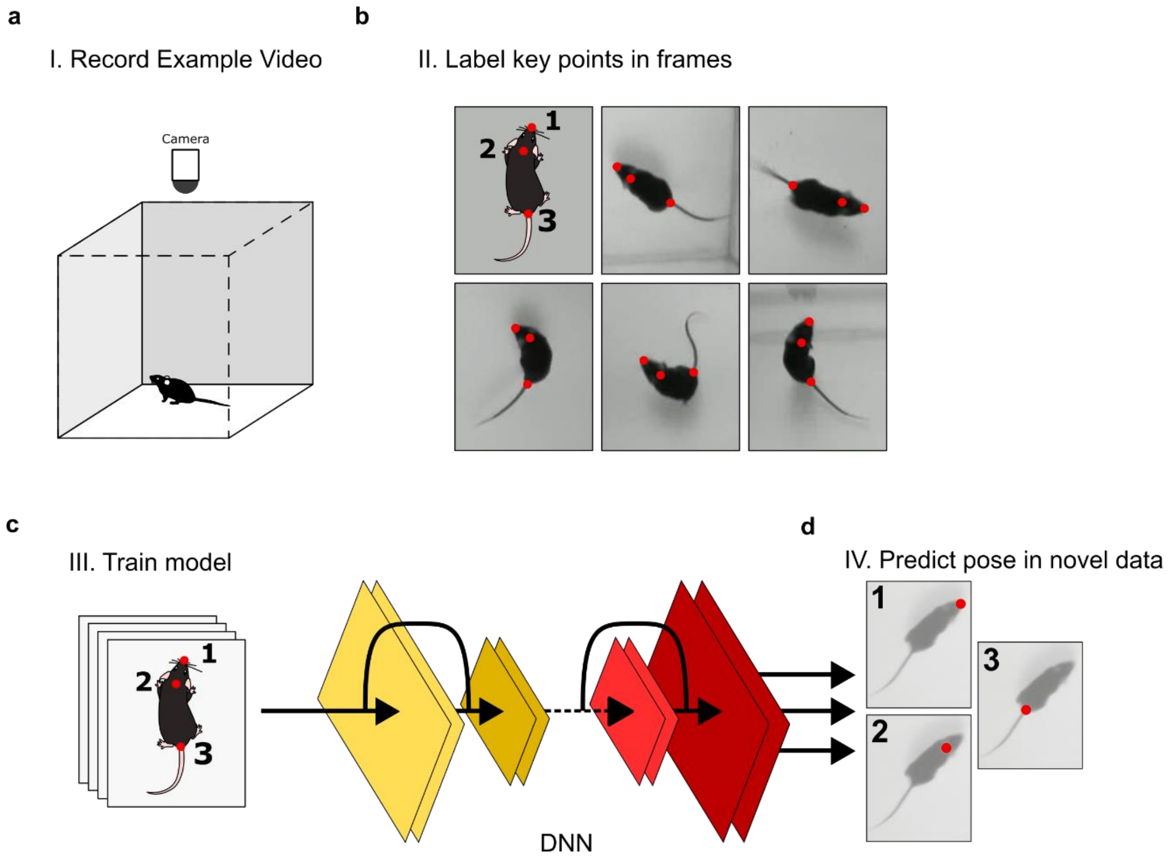


Figure 3 – Pose estimation using Deep Neural Networks

a, Schematic representation of an experimental setup. A camera, mounted above the arena at a 90° angle, is used to record example videos of a mouse during behavior.
b, A characteristic set of example frames recorded in **a** is labeled with key points (e.g., nose, neck, and tail base) and used to train a deep neural network (DNN; **c**).
c, The DNN extracts relevant image features and learns an abstract definition of the user-defined key points.
d, The trained network can then be used to estimate the position of previously learned key points in novel video frames. The resulting pose estimation is exported and can be used for complex behavior analysis. This figure was inspired by Mathis, Mamidanna et al. 2018.

Markerless pose estimation is one of the recent additions of machine learning-based approaches in ethology and neuroscience [4,39–41,43]. Its core achievement is the reliable, autonomous extraction of positional data of user-defined key points (e.g., body parts; Figure 3) from a video without the need for physical markers (e.g., reflective markers applied to the subject). This form of motion tracking has several advantages over classic videography, which is often a time-consuming and error-prone process. It allows additional degrees of freedom compared to marker-based tracking [72–74]: Primarily because markers do not need to be preset or predefined before the recording. Consequently, videos can be reanalyzed with different sets of key points depending on the analysis requirements. Previously recorded data can be revisited even if markerless pose estimation was not established in the laboratory during their recording. This advantage increases the likelihood of implementing markerless pose estimation in ongoing research projects successfully.

For deep learning-based pose estimation, a trained machine learning algorithm (model) identifies reoccurring features in a video frame and reliably extracts positional information with high accuracy (Figure 3). As a result of the growing interest from researchers across ethological fields in recent years, several implementations of this method are available for animal tracking [20,21,35,36]. Most recent popular models (e.g., DeepLabCut [39,47,48]) are based on deep learning architectures, such as DeeperCut[75], a model previously developed for human pose estimation. Deep neural networks (DNNs, Figure 3 c) utilize the inherent ability of artificial neural networks to learn how to extract high-level features from raw input, such as coordinates from video frames, based on previously seen labeled data (Figure 3 a-b). This ability allows researchers to quickly train a robust machine learning algorithm to identify body parts of interest in their experimental paradigm and track the movement of each key point across multiple sessions and individuals. Pose estimation data can then be analyzed in several ways, including machine learning-based approaches for behavioral classification [49,54,76]. The resulting behavioral classification can then be used to extract highly detailed information about the specific behavioral changes in each session (see chapter 3.3).

However, to study the activity and connectivity of neuronal networks underlying behavior, the respective behavioral expressions and corresponding neuronal ensembles need to be identified and labeled in real-time. This endeavor requires fast, reliable pose estimation and an advanced closed-loop system to identify behavioral expressions and administer real-time feedback.

3.3 AI-based behavioral analysis

With the rise of machine learning-based pose estimation of animals, the amount of readily available, highly detailed data on animal behavior is growing steadily. The demand for unbiased, high-throughput analysis resulted in several open-source applications that enable non-expert researchers to start analyzing their complex behavioral data [49,53–56,76,77]. Approaches range from classifying previously defined behavioral expressions [49,76] to finding novel patterns in the hidden dynamics of complex behavior [52–56,77]. These approaches often incorporate sequential analysis protocols for automatic parameter quantification [49,76] that previously required human expert annotation over hours of video data. Consequently, researchers who successfully established these toolkits benefit from the increased time efficiency, inherently low bias, and increased spectrum of complex behavioral expressions [49,53,54,76].

Considering closed-loop experiments, where behavior detection is often required to be autonomous and faster than humanly possible with minimal inter-event variability, a machine learning-based behavior analysis would increase the detection spectrum considerably. However, the practical benefit of using machine learning-based behavior detection should be considered on a case-by-case basis. Simple behavioral expressions can often be easily defined by relative feature changes and do not require the elaborate training of a classifier.

3.4 Light sheet fluorescence expansion microscopy

The ability to volumetrically image highly detailed molecular information in subcellular resolution across whole brain areas is critical in establishing meaningful, time-efficient studies of functional ensembles across the brain. With optical and electron microscopy, researchers established methods that have the potential to untangle the complexity of the brain's functional architecture. These neuronal architectures are composed of structures spanning several orders of magnitude across the brain. Unfortunately, optical microscopy is often insufficient to reveal subcellular details in high resolution, and electron microscopy lacks the molecular contrast to phenotype and investigate in rich detail over a larger scale.

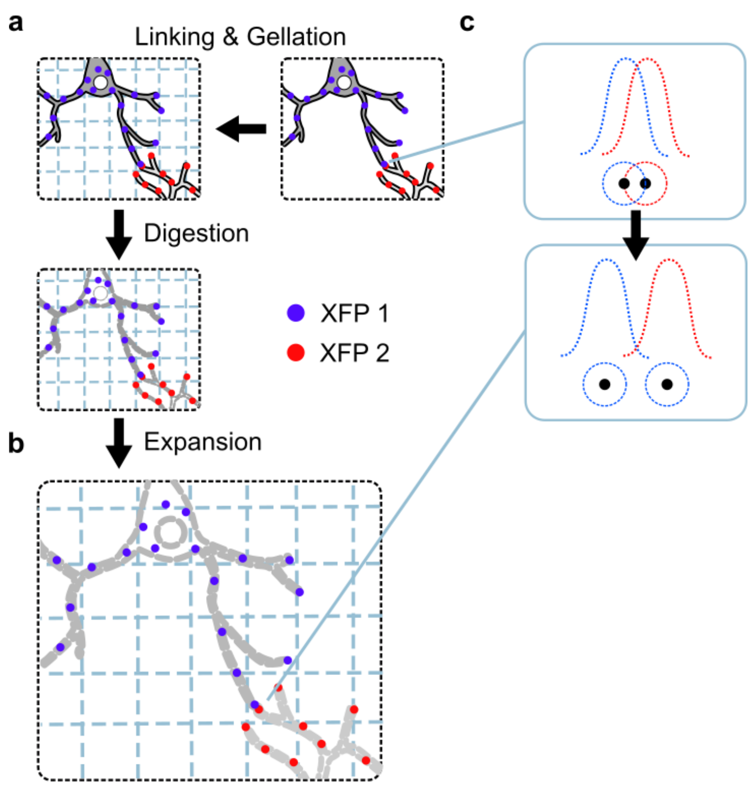


Figure 4 - Tissue expansion

a, Processing steps of tissue expansion for samples with fluorescent labels (XFP 1 and 2). First, a polymer matrix is formed by incubating the tissue in a gelling solution (see Methods), and fluorescent labels are covalently bound to the matrix (Linking). Digestion: the tissue is enzymatically cleared before expansion.
b-c, After expansion with water (**b**), fluorescent labels initially spaced closer than the optical diffraction limit (~250nm) can be optically resolved (**c**)

A solution built to combine super-resolution, high throughput, and high molecular contrast imaging of large brain samples is light sheet fluorescent expansion microscopy (LSFEM). Combining two optical microscopy techniques enables imaging from mesoscopic to nanoscopic scale [57,58].

Standard expansion microscopy virtually enhances the potential resolution of optical microscopy by increasing tissue sample size rather than optimizing microscopy techniques and equipment [57,58,78–80]. The tissue is permeated with a hygroscopic polymer and isotropically expanded after enzymatical treatment (see Figure 4). For this, proteins of interest are labeled with antibodies, nanobodies, or fluorescent tags (e.g., GFP), which are covalently attached to the polymer matrix before isotropic expansion with water-based solutions (Figure 4 a-b). After expansion, fluorescent labels initially spaced closer than the optical diffraction limit (~250nm) can be optically resolved, resulting in effective "super-resolution" images of the sample (Figure 4 c). Due to the high water content of the expanded sample, the tissue is rendered fully transparent, comparable to the results of chemical tissue clearing [81–85] but without the need for complex and potentially fluorescence damaging clearing protocols. Additionally, unlike chemical clearing, the tissue can now also be resolved in much greater detail. Notably, the increased size of the sample restricts the accessibility of deep structures with conventional super-resolution microscopy techniques.

The technique was recently combined with light sheet fluorescence microscopy (LSFM). In standard LSFM, samples are illuminated with a thin sheet of light, and emitted signals are detected orthogonally to the light sheet by wide-field detection [86] with a CMOS camera in a confocal line detection scheme [87,88]. LSFM is conventionally used for volumetric imaging of large, cleared samples at high speed but lacks the high resolution of other microscopy techniques [85,87,89–91]. As the synergetic combination of both techniques, LSFEM allows high-detail, large-scale volumetric imaging of synaptic connectivity maps in intact brain samples with high throughput [57,58].

3.5 Multicolor neuron labeling for circuit tracing

To fully characterize the functional architecture of neuronal circuits, it is essential to trace the connections of individual neurons within entire populations. However, most tracing techniques utilize methods that label neuronal populations in a single color resulting in considerable segmentation problems. While beneficial when studying general connectivity between brain regions, such an approach limits the ability to characterize the connectivity maps of neurons on an individual level. For example, in regions such as the hippocampal formation, neuronal populations have recurrent, widely distributed connections within a dense structure of layered neurons. To characterize such connections in great detail, the neuronal density needs to be countered with techniques that simultaneously minimize the potential loss of information. A famous example countering the segmentation problem in dense regions is sparse labeling [92], also utilized in the Golgi method [93,94]. However, while greatly reducing the overlapping of labeled neurons, connectomic analysis using this technique generally assume the stereometric homogeneity of neurons within a population and therefore likely neglect more complex differences within investigated populations.

A technique that utilizes stochastically distributed expression of multiple, differently colored XFPs, known as Brainbow [63–65], elegantly solves this challenge using a Cre/loxP-System [95–97]. In principle, Brainbow enables XFP expression in different levels across infected neurons. The resulting distribution of XFPs in different concentrations per cell

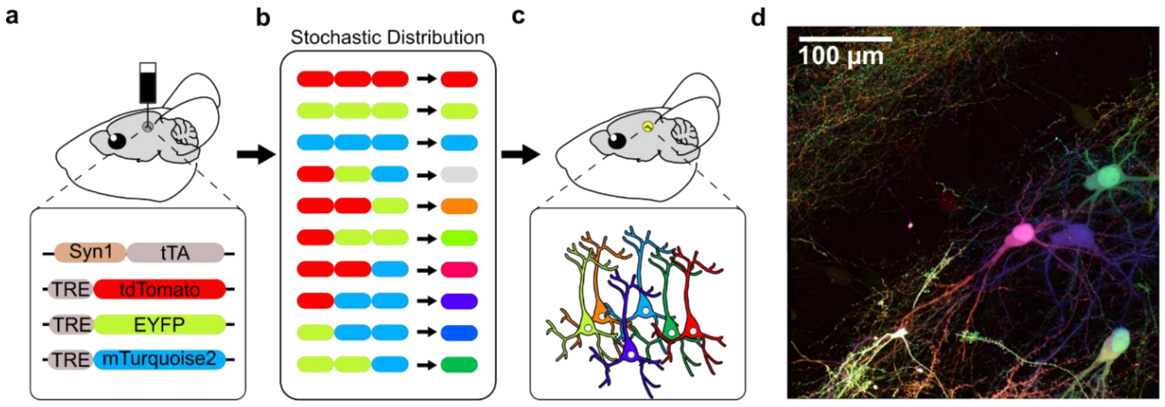


Figure 5 - Schematic representation of Tetbow

a, A virus mixture (AAV-Syn1-tTA, AAV-TRE-tdTomato-WPRE, AAV-TRE-EYFP-WPRE, AAV-TRE-mTurquoise2-WPRE) is injected into the target brain region containing expression vectors for the fluorescent proteins tdTomato (red), EYFP (yellow/green), and mTurquoise2 (blue).
b, The stochastic distribution during infection results in different copy numbers per infected cell, which will result in different color hues in RGB color space.
c, Neurons can be identified, and their extensions traced by their distinct color hue. The resulting color diversity is dependent on the total number of copies per infected cell.
d, Maximum intensity projection of a sample stack with Tetbow expression in the DG of the hippocampus after digestion treatment (see Methods 7.5). 4-tile stack acquired with the confocal Zeiss LSM880. Courtesy of Juan E. Rodriguez Gatica

results in a spectrum of color hues, effectively labeling neurons uniquely. However, resulting expression levels were often inadequate to detect axons and dendrites in large-scale tissue samples (e.g., using tissue clearing). Consequently, detailed connectomic analysis was often limited to the time-consuming, error-prone segmentation of thin serial sections. Recent advancement in multicolor labeling, Tetbow [62], raises expression levels high enough to allow whole-brain tracing in cleared samples and presumably expanded tissue (see chapter 3.4).

Tetbow achieves high expression levels and wide-ranging color hues by utilizing the Tet-Off expression system [98–100]. Tet-Off is derived from the tetracycline resistance operon in *E. coli*. Originally, the Tet repressor protein (TetR) inhibits transcription in the absence of tetracycline (an antibiotic) by binding to the *tetO* sequences in the promotor region. However, a eukaryotic transcriptional activator (tTA) was generated by fusion with a Herpes simplex virus VP16 trans-activator and combined with eukaryotic minimal promoters (TRE). The resulting expression system (Tet-Off) enables highly specific expression in the presence of tTA, while tetracycline addition results in strong inhibition of gene transcription. An additional advantage of the Tet-Off system lies in its wide use and the availability of a wide range of vectors that can be easily exchanged.

The original Tetbow system (see Figure 5 a) consists of a set of viral vectors with three different XFPs (TRE-tdTomato, red; TRE-EYFP, yellow/green; TRE-mTurquoise2, blue), which are expressed in the presence of the fourth vector (Syn1-tTA) in infected neurons. The resulting color hues (similar to the RGB color space, Figure 5 b-c) are spread across the visible spectrum by combining different fluorescent intensity levels based on the stochastic distribution of vector copies within each infected neuron [62]. Thus, the color diversity is directly linked to the distribution of XFP gene copies and tTA expression following a Poisson distribution [62,101]. However, it is important to note that the color diversity reduces as the number of introduced gene copies increases [62].

The resulting color diversity can also be used in a computational approach that segments neurons based on their unique color hue. A hue-based segmentation algorithm would increase tracing efficiency for automatized large-scale connectomics. Especially in combination with LSFEM, the ability to dissect individual neurons and resolve both long-range projections and highly detailed morphology across scales has the potential to fill the gap between wholesome but small-scale electron microscopy and large-scale tissue clearing in connectomics.

3.6 Vector-based delivery of expression systems

Stereotaxic delivery of recombinant adeno-associated viruses (rAAVs) is the go-to strategy for exogenic gene delivery in the postnatal rodent brain [102]. Its high precision and moderate invasiveness allow the temporally precise manipulation of gene expression in virtually any brain region and cell type, given the right combination of rAAVs. The technique is easily reproducible once the conditions for the desired gene expression are found. However, the initial adjustment of all parameters requires careful testing. With every new vector and brain region, the necessary amount of virus, the expression time until sacrifice, and optimal coordinates need to be considered.

The complexity grows accordingly when gene deliveries simultaneously require multiple rAAVs or vectors, as is the case for Cal-Light [26] and Tetbow [62] (see chapters 3.1 and 3.5). Specifically, the individual expression levels can vary drastically when using systems expressing multiple fluorescent proteins (XFPs) such as Tetbow with separate vectors in the same cell. Consequently, the resulting fluorescent intensity values render imaging all colors at equivalent levels a matter of careful fine-tuning for connectomic studies. Here, considering a range of mixtures is essential to finding the optimal combination of expression strength, viral spread, and cell toxicity.

For large-scale brain tissue, the injection volume and stereotactic coordinates are dependent on the targeted brain region and can be based on previous successful studies. However, the optimal mixture of different vectors is more complicated, especially when considering automatic segmentation, where the variability between samples should be minimal. With Tetbow specifically, there are four vectors to consider (see chapter 3.5). Briefly, the first vector acts as a general conductor for gene expression levels of the other AAVs, namely XFP expression. Notably, high expression levels increase the cell toxicity of viral delivery systems, so the overall viral load and expression levels need to be carefully adjusted. The other three vectors each express different XFPs (tdTomato, eYFP, mTurquoise) with individual fluorescent intensities and expression efficiency. Depending on the desired effect, the relative concentration of each vector needs to be adjusted. In the Tetbow-based approach for automatic segmentation, the ultimate goal is the unique labeling of neurons within a dense population. Consequently, the sufficient expression of all XFPs is required for a maximum range of available color hues [62]. This requires careful adjustments of all parameters – i.e., multiple surgeries - with the repeated evaluation of *post mortem* sections to find the right combination if an optimal expression profile is desired.

4 Results

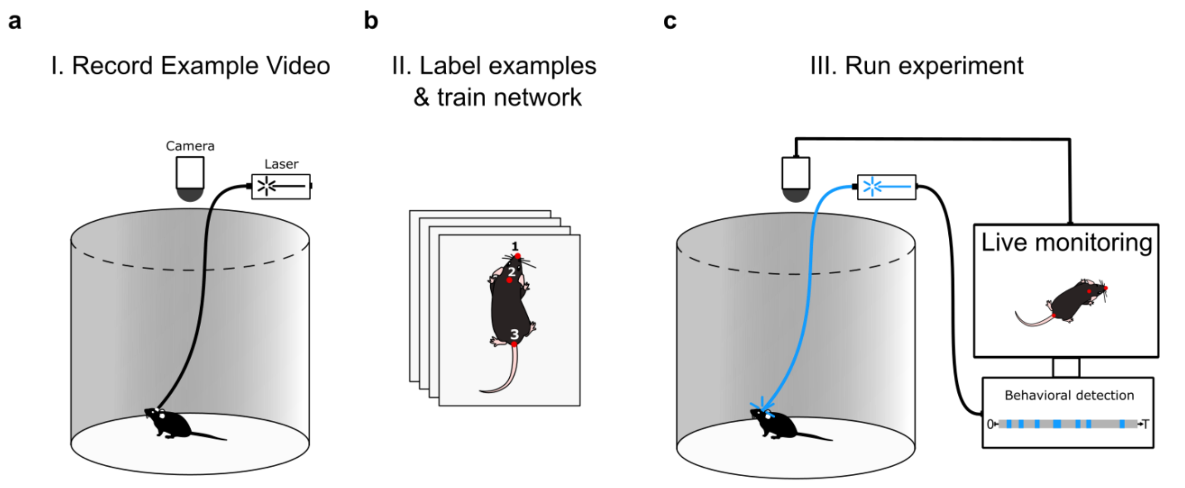
4.1 Real-time, closed-loop experiments

4.1.1 Real-time tracking and manipulation of animals during ongoing experiments

During the development of DeepLabStream (DLStream), the main goal was to create a software solution that enables closed-loop stimulations directly dependent on behavioral expressions. The resulting software is able to conduct behavior-dependent experiments fully autonomous and requires no additional tracking-, trigger-, or timing-devices. Primarily, experiments orchestrated by DLStream can be conducted without restriction to the animal's movement due to the optimized integration of real-time, markerless pose estimation. Additionally, DLStream was built so that input and output devices can be integrated freely into the hardware design of experiments (Figure 1).

For the conducted experiments, a pose estimation network was trained offline using DLC and was then integrated into DLStream (see 7.11 and 7.12). Briefly, frames of previous recordings of a mouse exploring the arena were taken and labeled as ground truth (Figure 6 a-b). The ground truth dataset was then used to train a deep neural network to recognize and estimate the positions of user-defined key points (neck, nose, and tail base).

In DLStream, frames taken from a camera stream positioned above the arena were analyzed using the integrated pose estimation network. The resulting pose information was converted into postures and transferred to an additional process. This process supervises the ongoing experiment and outputs feedback to connected devices (Figure 6 c-d). In principle, experiments run by DLStream comprise a sequence of modules depending on the underlying experimental protocol (Figure 6 d, Supplementary Information 9.3.2, Supplementary Table A, Supplementary Table B). Basic modules, such as *timers* and *stimulations*, are behavior-independent and control essential aspects of the experiments. *Timer* modules track time-dependent stages and act as a gate for behavior-dependent stimulation events (e.g., inter-stimulus timers). *Stimulation* modules specify which external devices are triggered and how each device is controlled once activated (e.g., optogenetic light stimulation; Figure 6 d). Behavior-dependent feedback is triggered by *trigger* modules that detect specified behavioral expressions. *Trigger* modules consist



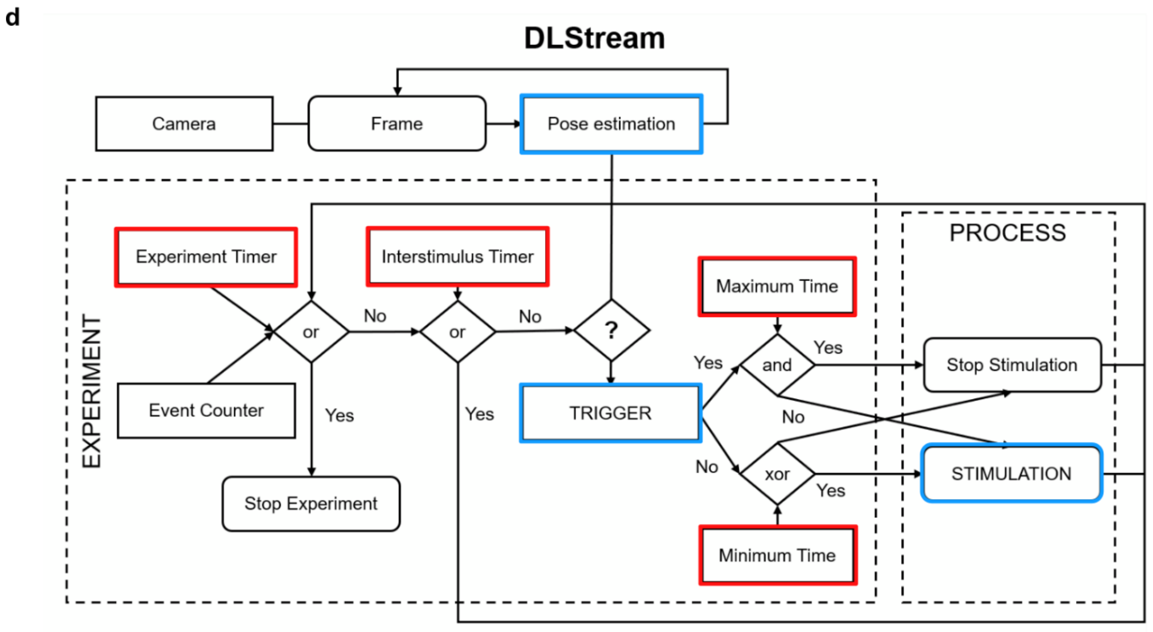


Figure 6 - Experimental setup

a, Schematic representation of a setup run with DLStream. A camera, mounted above the arena at a 90° angle, is used to record example videos of the arena, including a mouse with a fiber cord.
b, A set of example frames recorded in **a** is labeled and used to train a pose estimation network that can be integrated into DLStream.
c, Using the pose estimation of body parts (red dots), a behavior-dependent experiment is conducted with DLStream. Whenever DLStream detects a relevant behavioral expression (blue bars), the mouse is stimulated with light (blue cord).
d, Schematic representation of the underlying architecture for an optogenetic stimulation task. The sequence for behavior-dependent stimulation in **c** is highlighted in blue. Experiments run by DLStream typically incorporate time-dependent aspects controlled by timer modules (red) and consist of several logic gates (*and*, *or*, *xor*) to orchestrate essential aspects of the experiment. Any DLStream experiment is run as a loop on a frame-by-frame basis until a preset condition ends the experiment (e.g., the maximum number of stimulations or maximum duration).

of sets of defined postures (e.g., position, head direction) or are connected to behavior classifiers and initialize a predefined cascade (stimulation) once a behavioral expression was detected during an experiment (Figure 6 c-d, Supplementary Table A).

While experiments are conducted autonomously, ongoing experiments can be directly monitored on a live video stream visualizing pose estimation and experimental parameters (Figure 7). In DLStream, real-time pose estimation data, including relevant experimental parameters such as status, response latency, and event-onset, is exported as a table-based file (see chapter 4.1.2, Supplementary Table C). Additionally, the raw video camera stream during experiments is timestamped and recorded for *post hoc* analysis.

4.1.2 DLStream output

DLStream stores pose estimation data and information from experiments in a table-based file (Supplementary Table C). The table is saved as a CSV file that allows easy import into several software applications (e.g., Microsoft Excel®, MatLab®, Text editors, and Python-based analysis) across multiple operational systems.

The animal's position is linked to each frame by a frame-based index, additionally imprinted on the recorded video. In total, the output table provides information on the estimated position of all tracked body parts, the experiment's status, and a trial column used to give event/trial-specific information during experiments. Event-specific information can include different trials during conditioning or stimulation onset. The table also includes a time column where experimenters can see the exact inference time between each frame and the actual time passed during the experiment.

Like the raw pose estimation output from open-source solutions like DLC, the pose estimation and experimental data can be used for *post hoc* analysis.

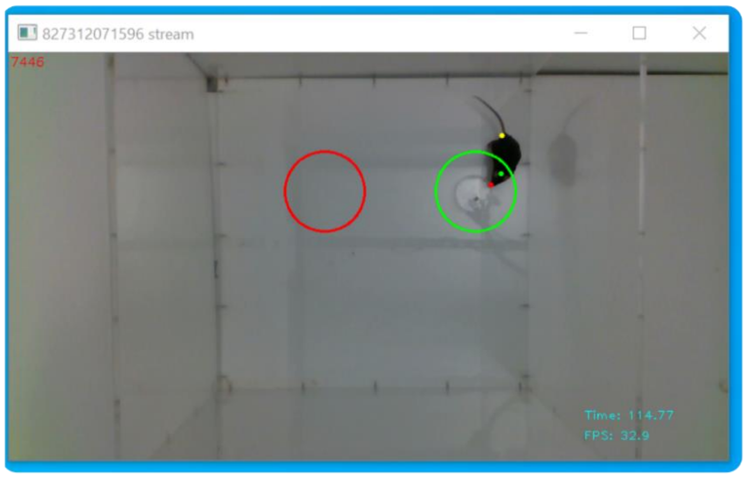
4.1.3 DLStream GUI

DLStream was developed so that non-expert users can conduct predesigned experiments without interacting with the underlying code architecture (see Supplementary Information 9.3.1). For this, DLStream has a graphical user interface (GUI) that can initialize, cancel, or finish up experiments (Figure 7).

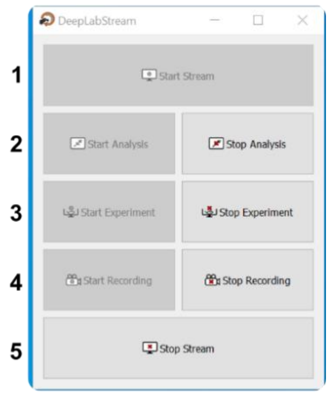
In a typical DLStream experiment, the video stream is started first (*Start Stream*, Figure 7 c1). The initialized live stream is then used to finalize the experimental setup (e.g., arena position, focus, and lighting conditions; Figure 7 a). Then, the pose estimation network is initialized (*Start Analysis*, Figure 7 c2), and its output is visualized on the live stream by colored dots (Figure 7 a-b). If desired, the live stream can be recorded (*Start Recording*, Figure 7 c3) and pose estimation data exported without an ongoing experiment.

However, starting the experiment will result in both automatically (*Start Experiment*, Figure 7 c4). During ongoing experiments, experimental information can be monitored on the live stream window (Figure 7 a-b) and console. At any point, users can stop DLStream, and collected data will be exported (*Stop …*, Figure 7 c1-5).

a



c



b

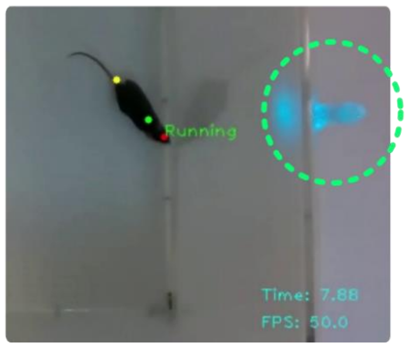


Figure 7 - DLStream Graphical User Interface

a, Example of a video live stream during DLStream conducted behavior experiment. The pose estimation on the nose, neck, and tail base are shown as colored points (red, green, and yellow). The results of two region of interest (ROI)-based trigger modules are visualized by the corresponding ROIs (colored circles). A positive detection is visualized in green (right, green circle), while a negative detection (mouse not in ROI) is visualized in red (left, red circle). The frame number since the beginning of the stream is imprinted on the video (top left, red), while the time since the beginning of the pose estimation (Time) and current latency (FPS) is shown as an overlay (bottom right, cyan).
b, Different example of the live stream shown in **a**. In this example, the mouse's locomotion was detected, indicated by the green text ("running") positioned at its nose. At the same time, an LED was activated by DLStream during the behavioral expression (green circle). A complete video version of this was published as a supplementary video in Schweihoff et al. 2021 [1].
c, Image of the graphical user interface of DLStream. Users can directly control DLStream using the buttons depicted.

4.1.4 Adaptability of DLStream

As with the development of a GUI, the goal for DLStream was to develop a software solution that can easily be customized and fitted to the experimental needs of several researchers. For this, DLStream's underlying code architecture was primarily built with modules that can be readily interchanged. The *stimulation*, *trigger*, and *timer* modules are encapsulated in an *experiment* module accessed by the main DLStream process (Figure 6 d). Briefly, any experiment running in DLStream follows the same logic (see Supplementary Information 9.3.2 and 9.3.5). Postural data is extracted from the incoming video frame by DLStream using a pose estimation network and passed to the *experiment*. Depending on the experiment's configuration, the posture will be passed to a *trigger* module that detects if the behavioral expression of interest was present. Independent of the type of *trigger* module, the output is a binary classification (True or False) typically used as an activation or deactivation signal for a *stimulation* module. Consequently, *trigger* modules are interchangeable by design and can be easily exchanged to customize existing *experiments*.

Typically, experiments are governed by behavior-independent parameters such as inter-stimulus times and fixed overarching paradigms (e.g., the maximum duration of an experiment and minimum stimulation time). *Timer* modules control such parameters and, similar to *trigger* modules, output binary information about ongoing timing. While *trigger* modules are typically designed for specific behavioral expressions, *timer* modules can be set, reset, started, and stopped as necessary within an *experiment* (Figure 6 d, Supplementary Information 9.3.2).

Preexisting *experiments* can be adapted by changing the underlying configuration or specific modules to create custom *experiments*. To facilitate the customization of experiments, the open-source published version [66] of DLStream includes step-by-step tutorials, several example *experiments*, and *trigger* as well as *stimulation* modules (see Supplementary Information 9.3, Supplementary Table A, Supplementary Table B).

Notably, DLStream experiments are not limited to a specific set of key points (body parts). They can utilize any combination of pose estimated body parts, even multiple animals in the same environment, independent of species. DLStream's posture data is stored as a skeletal representation (*skeleton*; a set of named body parts). Individual and sets of body parts can be selected to design *experiments* and *triggers* (Supplementary Information 9.3.5).

4.1.5 DLStream hardware compatibility

DLStream was successfully installed and tested on Windows 10 and Ubuntu 18.04.05 OS. The software was developed in the open-source programming language Python that includes open-source libraries for most available devices and desired functions. Consequently, DLStream can utilize and control a wide range of devices. Virtually any webcam/camera can be used with various framerates and resolutions, considering hardware requirements and pose estimation performance (see chapter 4.1.7). The current version of DLStream [66] can integrate cameras using the *OpenCV* library (generic cameras), *pypylon* (Basler cameras), *pyrealsense2* (Intel RealSense® cameras), and *pyzmq* (IP webcams). Notably, DLStream is also able to run with prerecorded videos. Using a simulated real-time video feed can be helpful to set up and design *experiments* and reduces unnecessary preliminary live sessions with animals to set up behavior detection.

Additionally, DLStream includes libraries that allow the *general-purpose input/output* (GPIO) control through three different device types: Dataaquisition boards from National Instruments (*nidaqmx*), Raspberry Pi boards (*pigpio, gpiozero*), and Arduino boards (*pyserial*). However, all devices are conveniently interfaced in DLStream, so that, independent of the device, the design of an experiment remains the same.

4.1.6 Optogenetic, head direction-dependent labeling of neurons using DLStream

The results of this experiment were also published in Schweihoff et al. 2021 [1].

The development of DLStream allowed the design of an experiment that would incorporate the possibility to label active neurons optogenetically depending on the behavior of mice. For this, an experiment was designed to label active neurons in the anterior dorsal nucleus of thalamus (ADN) dependent on the mouse's head direction using the neuronal activity-dependent labeling system Cal-Light[26].

To label ADN ensembles, light stimuli were delivered within precisely defined head direction angles (target window) (Figure 8 a-b). Using DLStream, the onset and offset of light stimulation was controlled with *timer*, *stimulation*, and *trigger* modules as previously described (Figure 6 c, Figure 8 b; Supplementary Information 9.3.2). Mice were placed in a circular white arena with a single black cue at one side. The arena was kept in the same orientation throughout the whole experiment to ensure stable angular tuning. During the experiment, mice investigated the arena in one 30-minute session per day for four consecutive days (Figure 8 b). During each session, the mice were stimulated via a chronically implanted optical fiber with blue light (488 nm) depending on their head direction angle. The head movement of the mice was not restricted, and mice moved their head freely in all directions (Figure 8 a-c, Supplementary Figure A). During each session, mice explored the entire arena without restriction (Figure 8 e-f). However, light stimulation was limited to periods when they oriented their head to the target head direction window (60° to reference point; Figure 8 b-c, Supplementary Figure A). Each stimulation lasted 1-5 sec depending on the time spent orienting to the target window with a minimum inter-stimulus time of 15 seconds. During inter-stimulus periods, a *timer* module blocked the link between the *trigger* and *stimulation* module, disabling behavior-dependent stimulation for its designated duration (Figure 6 c, Figure 8 b).

The average light stimulation per session was 48 ± 10 seconds and occurred selectively in the target angle window across all experimental animals (Figure 8 h). Notably, light stimulation outside of the target head direction window can result from the preset stimulation conditions. Each stimulation was set to a minimum duration of 1 second, in which mice were able to sweep their head out of the target window. Nevertheless, the average total stimulation time across all four sessions was 357 ± 53 sec (n = 10 mice) with a significantly higher stimulation in the target window (Figure 8 h). Analogously, head direction-specific light stimulation could not have been achieved by random stimulation during the session. A random sampling of observed head direction angles equal to the number of stimulation events in individual sessions revealed a nonspecific distribution of covered angles – i.e., mice oriented in all directions (Figure 8 d, left).

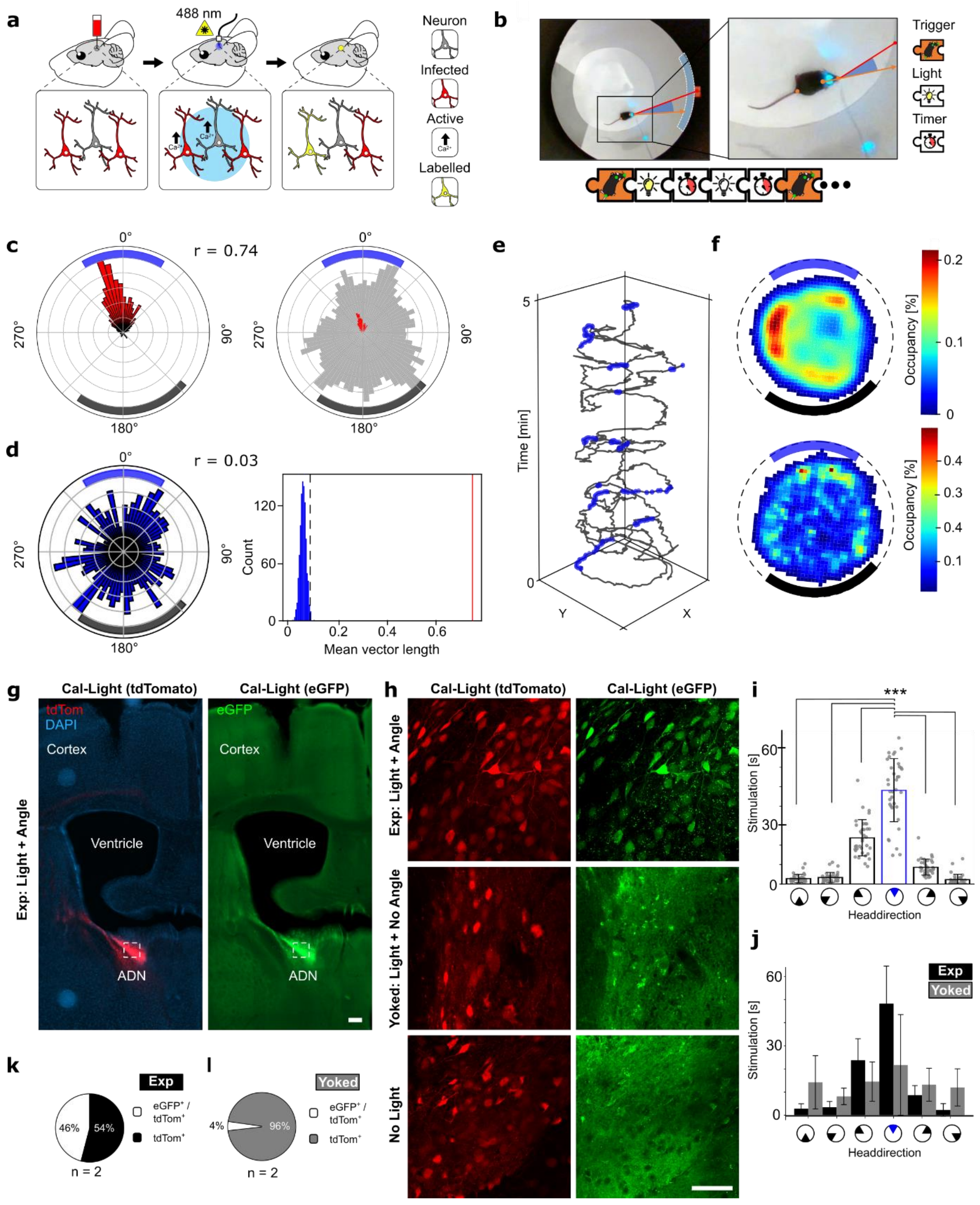


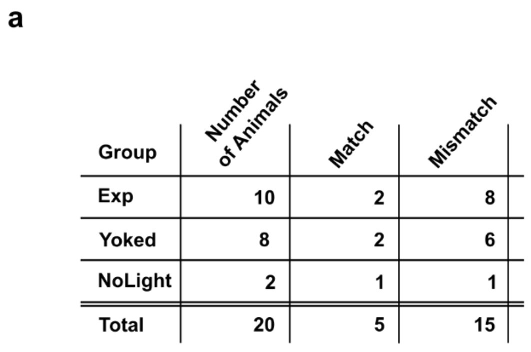
Figure 8 - Optogenetic labeling of head direction-dependent neuronal activity

a, Left: Stereotactic delivery of Cal-Light viruses into the ADN and fiber ferrule placement. Middle: Infected neurons (red) are stimulated with blue light (488 nm) controlled by DLStream. Right: infected neurons are only labeled (yellow) when they are active (black arrow up) during light stimulation (middle). **b,** Example images of head direction-dependent light stimulation. The mouse's pose estimation (orange dots) is used to calculate its head direction angle (orange arrow) related to a reference point (red line). Light stimulation is triggered if the head direction angle is within the target window (blue arc). A schematic representation of the sequence of modules (puzzle pieces) used in the design of this experiment is shown beneath the images (see also Figure 6 c). Timer modules are used as a minimum stimulation timer (left) and an inter-stimulus timer (right).

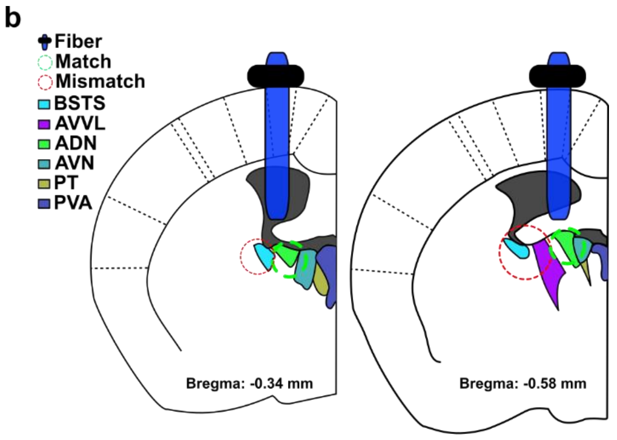
c, Left: Representative example (see also Supplementary Figure A) radial histogram of all head directions during stimulation (red) within one session (normalized to the maximum value). Mean resultant vector length is indicated by r. Right: Radial histogram of all head directions during the whole session (grey) and stimulation (red) The values were normalized to the maximum value of the entire session. Rings represent quantiles in 20 % steps.
d, Left: Representative random sample of covered angles during the whole session simulating random stimulation. Simulated stimulations are triggered without DLStream control at random time points during the session (normalized to the maximum value). The mean resultant vector length is indicated by r. For each session, random distributions were calculated 1000 times. Right: The distribution of mean resultant vector lengths generated by random sampling (n = 1000) of a single session. The red line denotes the actual mean resultant vector length during stimulation in the same session. The dotted black line represents the p<0.01 cutoff.
e, Representative example of the mouse's position (grey) over time during the first 5 minutes of the session in **c**. The stimulation events are shown in blue.
f, Heatmaps representing the relative occupancy of the mouse within the arena during the whole session (top) and only during stimulation events (bottom) in **c**. The cue and target window are shown in their relative position.
g, Example of Cal-Light expression in a mouse from the experimental group. Left: tdTomato expression (red) indicating expression of Cal-Light viruses with nucleus staining (DAPI, blue). Right: Activity-dependent and light-induced eGFP expression (green) in the same section. The white box represents the zoomed-in region in **h**. The bar represents 200 µm.
h, Close up from **g** vs. a similar region in an animal not stimulated with light (No Light group) and in the yoked control group. Left: tdTomato expression (red). Right: Activity-dependent and light-induced eGFP expression (green). The bar represents 50 µm. Note that control mice show no eGFP expression in tdTomato$^+$ neurons. In the yoked group, light stimulation of the same duration as in the experimental group but not the same head direction specificity did not result in sufficient Cal-Light labeling.
i, Average light stimulation during each session (40 total) corresponding to head direction (60° bins) with target window (blue wedge) indicating the DLStream triggered stimulation onset in the experimental group. Paired student's t-test: p < 0.001. n = 10 mice. Error bars represent standard deviation.
j, Average light stimulation in both experimental and yoked groups during each session as a function of head direction (60° bins) similar to **i.** Exp: n = 10 mice, black bars; Yoked: n = 8 mice, grey bars. Error bars represent standard deviation. Experimental and yoked groups have the same total stimulation time, but the distribution differs such that the yoked group has approximately equal stimulation times across varying head direction angles.
k, Ratio between infected neurons (tdTom$^+$) and activity-dependent labelled neurons (eGFP$^+$/tdTom$^+$) in mice matching selection criteria (see Methods). n = 2 mice.
l, Ratio between infected neurons (tdTom$^+$) and activity-dependent labeled neurons (eGFP$^+$/tdTom$^+$) in mice matching selection criteria (see Methods) in the yoked group. n = 2 mice.
A modified version of this figure was also published in Schweihoff et al. 2021[1].

Note that for each mouse, the mean resultant length for stimulated angles was significantly larger than would be expected by random sampling (see Methods, n = 1000 samples, p < 0.01) (Figure 8 d, right).

As an additional control, a yoked group of mice was run such that each mouse, regardless of its actual head direction, received the exact same temporal stimulus as a paired experimental mouse. Therefore, in the yoked group, light stimuli were decoupled from the individual head direction (Figure 8 j). Thus, in theory, if neurons are inconsistently



Group	Number of Animals	Match	Mismatch
Exp	10	2	8
Yoked	8	2	6
NoLight	2	1	1
Total	20	5	15



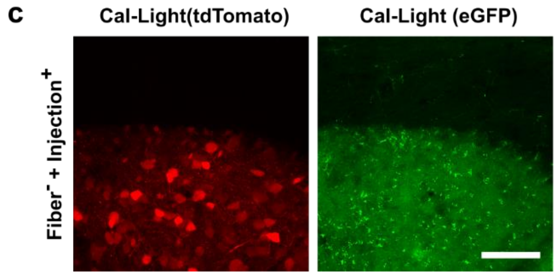


Figure 9 - Quantification of optogenetic labeling of head direction-dependent activity in neurons.

a, Table of all injected and implanted animals divided into experimental groups and success categories. 'Match' occurred when the viral injection was successfully targeted to the ADN and optic fiber was placed above the ADN. Tissue processed from the 'match' case was used for the quantification of labeled neurons. 'Mismatch' occurs either when the viral injection or fiber placement missed the ADN.
b, Schematic representation of injection sites. When the ADN was missed, injections were too lateral, hitting either the BSTS or AVVL. The blue ferrule represents optimal placement of the light fiber.
c, Close up (similar region of interest as shown in Fig. 4g) of representative expression in mice with incorrect fiber placement. Left: tdTomato expression (red) indicating expression of Cal-Light viruses. Right: Activity-dependent and light-induced eGFP expression (green). The bar represents 50 µm.
BSTS: bed nucleus of stria terminalis, supracapsular part; AVVL: anteroventral thalamic nucleus, ventrolateral part; ADN: anterodorsal thalamic nucleus; AVN: anteroventral thalamic nucleus; PT: paratenial thalamic nucleus; PVA: paraventricular thalamic nucleus, anterior part.
A version of this figure was also published in Schweihoff et al. 2021 [1] as Supplementary Material.

active during all stimulations, the head direction independent stimulation should result in insufficient Cal-Light labeling of head direction correlated activity.

The percentage of Cal-Light labeled ADN neurons was quantified in the three different groups (experimental, no-light, and yoked). Initially, a group of 20 animals was injected with Cal-Light, implanted with a light fiber, and used for the experiment. However, after confirmation during *post mortem* analysis, only mice that showed correct fiber and injection placement were considered for labeling quantification (see Methods and Figure 9 for details). Mice excluded from the quantification were still included in the evaluation of DLStream performance.

Cal-Light infected neurons showed a 46 % conversion within the ADN (Figure 8 k, n = 2 mice), while mice receiving no light stimulation but underwent the same sessions had no light-induced labeling present (Figure 8 g-l). Furthermore, within the yoked group, only a very low percentage (~4 %, n = 2 mice) labeling was observed (Figure 8 j, l). This indicates that light stimulation of the same duration as in the experimental group but not the same head direction specificity was insufficient to activate the Cal-Light labeling system reliably.

4.1.7 Computational performance of DLStream

The results of this evaluation were also published in Schweihoff et al. 2021 [1].

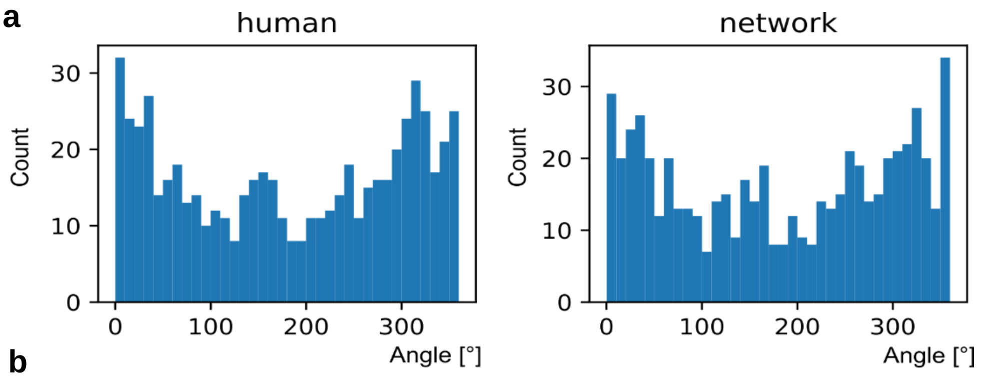
A reality of any closed-loop system is that there are temporal delays between real-time detection of behavioral expressions and stimulus output resulting in potential inaccuracies that need to be within acceptable margins.

Fundamentally, the variance of extracted behavioral parameters is dependent on reliable pose estimation. The pose estimation error of the applied model and the correlated parameter changes between frames need to be compared to estimate the spatiotemporal resolution of detectable postures. Due to the inherent individual model performances, DLStream's effective accuracy in posture detection is heavily influenced by the previous training of utilized pose estimation networks. Nevertheless, if performance is not sufficient for the executed experiment, deep neural networks can consistently be retrained using the respective open-source tools. The trained DLC model used during optogenetic experiments had an average pose estimation error of 4 ± 12 pixels (px) for the neck point, 3.3 ± 4.4 px for nose, and 3.3 ± 2.0 px for the tail base (n = 597 images) when compared to a human annotator labeling the same data set. For reference, mice without tails were ca. 60 px long in 848x480 px video recordings. Consequently, body part estimation resulted in an average head direction variance of 3.6 ± 9.6° (tested in 80 sessions for 1000 frames per session) between consecutive frames with an estimated average error of 7.7 ± 15.1° compared to human annotation (n = 597, ground truth) per frame. The frame-by-frame variance is a product of performance errors and the inhomogeneous movement of the animal during experiments. At the same time, the difference between network pose estimation and human annotation is most likely a result of inaccurate tracking, which can be reduced by additional training and more extensive training sets. Note that this variance might change depending on the mixture of episodes of fast movements and slow movements during sessions. While DLStream's effective implementation depends on the integrated model's accuracy, the general suitability of the software should primarily be evaluated independently of the pose estimation accuracy as researchers deploying DLStream will have to train their own pose estimation network.

Manual evaluation of behavior detection accuracy during optogenetic experiments showed a false-positive rate of 11.8 % – i.e., activating a light stimulation without the mouse's head direction being in the target window. In the evaluated sessions, most false-positive events were anomalies in mouse behavior prone to pose estimation errors, such as spontaneous jumping. Inaccuracies like this can be further reduced by additional model training if necessary. Additionally, similar results were achieved based on a human-labeled data set (n = 597). The estimated general false-positive/false-negative rate for the configured head direction trigger was 11.1 ± 4.1 % (false-negative) and 11.6 ± 4.8 % (false-positive; Figure 10).

During the optogenetic experiment (n = 80), DLStream reached an average performance time of 33.32 ± 0.19 milliseconds per frame, matching the average camera framerate of 30 Hz (33.33 milliseconds). The performance time includes pose estimation, behavior detection, and computation of experimental protocols up to the final output. Additionally, hardware latency was measured to estimate the time between behavior detection and triggered stimulation during optogenetic sessions from three different mice (n = 164 stimulation events). The resulting light stimulation occurred within five frames (4.8 ± 1.1 frames at 30 fps; ≈ 150 ms). Notably, the total latency critically depends on the individual setup and the intrinsic parameters of connected components. In experiments requiring faster output, the setup can be further optimized to reduce hardware latency.

a



b

Window size [°]	False positive detection [%]	False negative detection [%]
60	11.6 ± 4.8	11.1 ± 4.1
50	13.2 ± 5.3	13.7 ± 5.4
40	14.7 ± 5.8	14.7 ± 5.9
30	20.1 ± 11.2	19.8 ± 10.5
20	29.0 ± 20.7	28.2 ± 18.7
10	72.8 ± 75.6	75.8 ± 89.0

Figure 10 - Estimation of accuracy of head direction triggers with different angle window sizes.

a, Histograms (10° bins between 0-360°) of the distribution of the labeled dataset (n = 597), with human annotation (right) and head direction angle based on network pose estimation (right) using the network trained for the optogenetic stimulation task.
b, Table showing the network pose estimation's false-positive and false-negative detection rate against human annotation in several differently sized angle windows (simulated triggers). The window was moved around in steps to counter any effects of non-uniform distribution. The average, as well as the standard deviation, were taken from all detected events. An event was counted as false-positive if the pose estimation resulted in a head direction within the window, while the human annotation did not (and vice versa for false-negative).
A version of this figure was also published in Schweihoff et al. 2021 [1] as Supplementary Material.

Different hardware configurations were tested to evaluate the limits of DLStream, including performance levels and response time. First, average performance was measured during 10000 frames in two different configurations with two different camera settings (30 fps and 60 fps with 848x480 px resolution) using the same camera used in the optogenetic experiment. With the standard 30 fps camera setting, the advanced configuration (Intel Core i7-9700K @ 3.60 GHz, 64 GB DDR4 RAM, and NVidia GeForce RTX 2080 Ti (12 GB) GPU) achieved reliable 30 fps (33.33 ms per frame) real-time tracking with 30 ± 7 ms inference time. The other system (Intel Core i7-7700K CPU @ 4.20 GHz, 32 GB DDR4 RAM, and NVidia GeForce GTX 1050 (2 GB) GPU) only reached an average analysis time of 91 ± 10 ms. Using a higher framerate input from the camera (60 fps; 16.66 ms per frame), the overall performance did not change drastically (24 ± 9 ms and 90 ± 9 ms, respectively). To address camera-specific limitations, a different camera was tested

Table 1 - Performance of different network architectures in DLStream in relation to the number of estimated body parts and image resolution.

Network	**Resolution**	**Average fps**	**Average fps**	**Average fps**
MobileNetv2	**320x256**	**164.04 +/- 7.28**	**130.55 +/- 6.51**	**79.29 +/- 19.18**
	416x341	119.73 +/- 8.42	86.64 +/- 3.43	67.50 +/- 10.95
	640x512	60.51 +/- 2.01	54.24 +/- 0.94	46.76 +/- 2.91
	1280x1024	16.61 +/- 0.26	16.19 +/- 0.20	14.99 +/- 1.14
ResNet50	320x256	107.58 +/- 8.68	94.30 +/- 6.21	67.01 +/- 10.36
	416x341	79.52 +/- 3.00	66.70 +/- 1.83	55.49 +/- 4.99
	640x512	44.92 +/- 1.44	41.03 +/- 0.52	36.50 +/- 1.88
	1280x1024	13.61 +/- 0.35	13.25 +/- 0.12	12.32 +/- 0.86
ResNet101	320x256	68.29 +/- 2.23	64.72 +/- 1.86	60.35 +/- 6.13
	416x341	54.85 +/- 1.51	49.99 +/- 0.94	48.34 +/- 3.01
	640x512	32.05 +/- 0.51	30.47 +/- 0.33	30.18 +/- 1.88
	1280x1024	9.80 +/- 0.28	10.33 +/- 0.34	9.92 +/- 0.76

(Basler acA1300 – 200 um), which lacks the depth capabilities of the Intel RealSense camera but comes with an increased framerate. DLStream's upper-performance limits were benchmarked with more standardized resolutions (ranging from 1280x1014 to 320x256 px) on the advanced configuration using the new camera. The initially trained DLC model used in the optogenetic experiment was based on the ResNet50 [75,103] architecture. However, several configurations and models were tested as well to get an overview of the other available models (ResNet101 [75,103], MobileNetv2 [104]) and a higher number of body parts (3, 9, and 13). During this benchmark, DLStream's latency reached a maximum of 130 ± 6 Hz (ca. 8 ms) with the MobileNetv2 architecture at 320x256 px resolution. In contrast, the ResNet50 network reached its upper limit at 94 ± 6 fps (ca. 10 ms) at the same resolution (see Table 1 for more details).

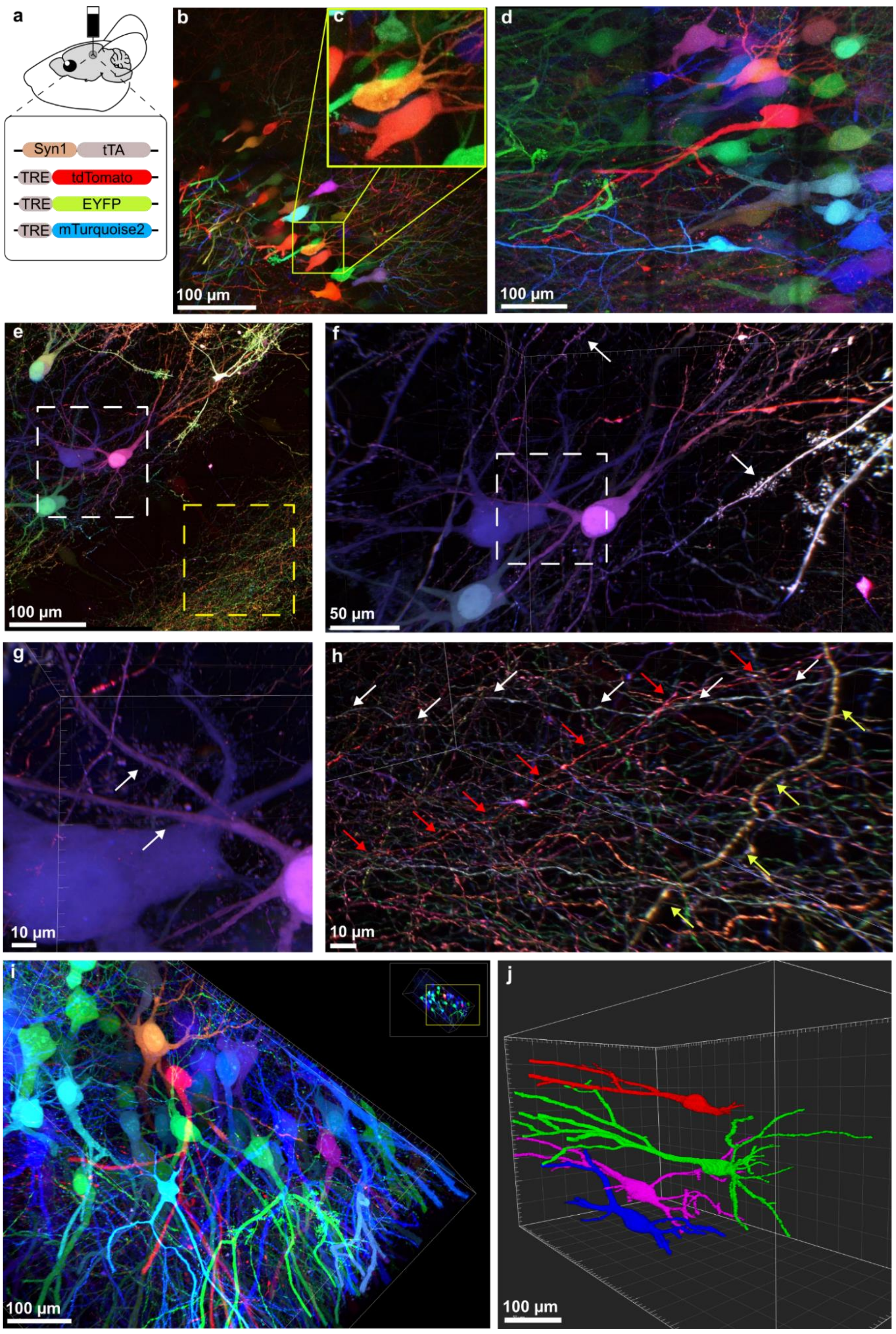
4.2 Multicolor labeling for neuronal tracing

Combining powerful imaging techniques such as confocal imaging and LSFEM with high-contrast biomolecular tools paves the way for scale-bridging connectomics. With conventional methods, the efficient identification and tracing of multiple individual neurons within a population is limited by the ability to distinguish between closely neighboring cells. A limitation that is especially important to consider in a brain region such as the hippocampal formation, where the densely layered neuronal architecture results in bundled axonal projections and heavily entangled dendritic trees. Here the power of high contrast, multicolor labeling using Tetbow [62] allows the distinction of closely neighboring neurons.

However, Tetbow was not yet shown to work with tissue expansion protocols and virtual super-resolution imaging. In close collaboration with the Institute for Theoretical and Physical Chemistry at the University of Bonn, the following preliminary study was conducted to implement this type of analysis for the automatic segmentation of large-scale connectomic studies bridging population-level analysis and single neuron tracing.

To facilitate the high contrast, high-resolution imaging of Tetbow-labeled samples, mice were injected with the Tetbow vector in different concentrations to evaluate the effective expression and resulting color variations in the selected region (see Supplementary Table D). For this, the parameters for optimal color variation and volumetric spread needed to be adjusted. A sample series was produced covering the most reasonable combinations following repeated *post mortem* evaluation of resulting XFP expression levels between each new batch of surgeries (see Supplementary Table D).

The resulting XFP expression was evaluated based on the signal intensity in all three channels (eYFP, tdTomato, and mTurquoise) and the observed color variation (see Supplementary Table D). Selected samples were then expanded and imaged with a custom build LSFM [105] (Imaging: Dr. Jana Heysel; Expansion and Imaging: Juan E. Rodríguez) in the collaborating laboratory of Prof. Dr. Ulrich Kubitscheck (Figure 11 i, see Methods 7.6). The custom LSFM allowed the high-speed, virtual super-resolution imaging of expanded samples multicolor-labeled with Tetbow. Imaged samples were evaluated before and after digestion to evaluate the effectiveness of Tetbow in combination with expansion microscopy and LSFEM (see Methods 7.5). Digestion and expansion of tissue samples resulted in minimal to no observable fluorescence intensity loss, allowing the direct imaging of samples using high-resolution imaging techniques (Figure 11 b-d). Notably, individual neurons can be identified by the naked eye by their unique color hue



a
Syn1
tTA
TRE
tdTomato
TRE
EYFP
TRE
mTurquoise2
b
c
100 µm
d
100 µm
e
100 µm
f
50 µm
g
10 µm
h
10 µm
i
100 µm
j
100 µm

Figure 11 – High contrast, multicolor labeling with Tetbow and tissue expansion for neuronal tracing

a, A virus mixture (AAV-Syn1-tTA, AAV-TRE-tdTomato-WPRE, AAV-TRE-EYFP-WPRE, AAV-TRE-mTurquoise2-WPRE) is injected into the target brain region containing expression vectors for the fluorescent proteins tdTomato (red), EYFP (yellow/green), and mTurquoise2 (blue).
b, Maximum intensity projection (MIP) of the CA3 area in the hippocampal region, injected with Tetbow vector. Imaged with LSM Zeiss 880 40x/NA 1.1. Courtesy of Juan E. Rodriguez Gatica.
c, Zoom in of region in **b** (yellow square) showing two closely neighboring neurons that can be discriminated by their distinct color hue. The hue results from the stochastic distribution of viral vectors and different expression levels of the XFPs.
d, MIP of same sample and region as in **b** after digestion (see Methods 7.5). The fluorescent intensity is conserved, and neurons, as well as neurites, remain distinct. Imaged with LSM Zeiss 880 40x/NA 1.1. Courtesy of Juan E. Rodriguez Gatica.
e, MIP of an image stack of the DG area in the hippocampal region after digestion. The white box represents the regions shown in **f-g**. The yellow box represents the region shown in **h**. Imaged with LSM Zeiss 880 40x/NA 1.1. Courtesy of Juan E. Rodriguez Gatica.
f, 3D view of a 4-tile image stack shown in **e**. Neurites running closely along other neurons can be discriminated by their color hue (white box, **g**). At the same time, the imaging resolution allows the discrimination of small details such as neuronal protrusions (arrow). The image stack has a size of 595.35 x 403.84 x 402.48 µm^3.
g, Zoom in of region in **f** (white box). The achievable broad hue spectrum allows to differentiate neurites running closely along neighboring neurons, including spine-like protrusions (arrows).
h, Zoomed in 3D view of a 4-tile image stack shown in **e** (yellow box). Tetbow expression is strong enough to allow consistent volumetric labeling across neurites. The unique hue allows tracing multiple neurites across the entire field of view (colored arrows) in a densely bundled region.
i, 3D view of an image stack acquired with a LSFM (see Methods 7.6) of a similar sample as **b** after digestion. Courtesy of Dr. Jana Heysel [105].
j, Representative example of hue-based tracing with Tetbow. Individual neurons can be identified and segmented based on their unique color hue. Courtesy of Dr. Jana Heysel [105].

(Figure 11b-e, i), and automatic segmentation allows the highly detailed analysis of individual neurons.

Slight expansion of treated samples with PBS (1.2-1.5 expansion factor; approx. 400 µm sample thickness) facilitates the distinction of closely neighboring neurons, including neurites and spine-like neuronal protrusions (Figure 11 e-g), while still allowing high-resolution imaging with conventional confocal microscopes. Here, the preserving nature of enzymatic clearing utilized in tissue expansion increases the available imaging depth considerably.

Additionally, the high expression of XFPs using the Tetbow system facilitates volumetric labeling of long-range connections. Thus, neurites within fiber bundles can be traced individually by their unique color hue across the entire field of view (approx. 600 µm; Figure 11 e, h).

Imaging data was then used to develop an algorithm to segment neurons based on their unique color hue [105] (developed by Dr. Jana Heysel). An example of the segmentation can be found in Figure 11 j. The results of this segmentation are published elsewhere [105].

5 Discussion

The primary goal of this thesis was the establishment of neuroscientific tools that allow the disentanglement of the complex relationships between neuronal ensembles and correlated behavioral expressions. For this, I utilized the biomolecular labeling system Cal-Light [26], which allows the light-induced, activity-dependent labeling of neurons in the rodent brain. To elevate the system's functionality to capture the behavior-dependent activity of freely moving mice, I developed an open-source, closed-loop experimental toolkit, DLStream, that enables the real-time detection of behavioral expressions and consequent orchestration of behavioral experiments. As such, DLStream enables the behavior-dependent light stimulation of animals in optogenetic experiments. As a proof-of-concept, I combined both tools in an optogenetic experiment to label neuronal ensembles active during specific head directions. The results of this experiment were also published alongside another experiment show-casing DLStream's capabilities elsewhere [1]. The toolkit was developed with sustainability in mind and was published open-source to facilitate closed-loop experiments in the neuroscientific community [66].

Finally, this thesis elaborates on the collaborative approach to increase the capabilities of the currently established connectomic analysis. For this, I established the biomolecular, multicolor labeling system Tetbow [62], which allows the unique labeling of individual neurons within a dense population of labeled cells. However, the effective use of Tetbow requires accurate tuning of several parameters due to its expression dynamics. After extensive testing, produced samples were subsequently expanded and imaged using LSFEM, enabling fast, super-resolution imaging of large-scale tissue [57]. The combination of Tetbow and LSFEM enables the identification of large-scale structures with a concurrent resolution to study individual connections on a synaptic level. To facilitate the large-scale connectomic analysis, the imaged data was then used to develop an automatic segmentation algorithm that utilizes the unique labels provided by the Tetbow approach to segment neurons in large tissue samples [105]. Representative examples of this collaboration are shown in the result section (chapter 4.2).

5.1 Real-time, closed-loop experiments

To investigate the neuronal correlates of complex behavior, it is necessary to identify and manipulate actively participating neuronal ensembles [5,7,8,106]. Therefore, techniques that bridge connectomics, electrophysiology, and ethology hold the potential to reveal how computations are realized in the brain and subsequently implemented to form behavioral expressions. For instance, by utilizing neuronal activity-dependent labeling systems such as Cal-Light [26], Flare [28,29], or CaMPARI [107], it is possible to capture neurons active during selected behavioral expressions. However, a lack of dynamic closed-loop systems restricts the reliable detection of reoccurring behavioral expressions and subsequent real-time feedback.

With the development of DLStream, the range of detectable behaviors increases substantially, and applications for behavior-dependent labeling and subsequent manipulation of different freely moving species are wide-ranging.

5.1.1 Head direction-dependent labeling of active neuronal ensembles

In this thesis, DLStream was used to orchestrate behavior-dependent light stimuli to the ADN and label neural ensembles active during specific head directions. Notably, the ADN was selected because it fulfilled two requirements. First, activity within ADN neurons is known to be modulated by the head direction angle [108–111]. Thus, the angular tuning curve of these neurons remains constant in stable environments – i.e., the same neurons will be active within the same head direction angles if the mouse is put in the same environment. This stability facilitates experimental designs that span several days [108], including the repeated stimulations necessary for high-contrast labeling with Cal-Light. Second, the ADN's structure is convenient for optogenetic applications. Due to its compactness, viral solutions injected into the ADN can spread through a large portion of the nucleus. At the same time, illumination through an implanted light fiber will evenly cover most of the infected region.

Head direction offers several advantages as a showcase for behavior-dependent, optogenetic labeling. Foremost, a causal relationship between head direction and neuronal activity within the ADN was established [108,110–112]. Thus, labeling active neuronal ensembles will most likely capture head direction cells, while Cal-Light's labeling requirements are likely to filter out head direction-independent activity. Additionally, the behavior can be easily tracked with pose estimation but is too fast for a human observer to identify in real-time reliably.

In practice, head direction-dependent light stimulation of active neurons was successful and resulted in eGFP expression in a subset of Cal-Light infected cells in experimental mice (ca. 46 %, Figure 8 g-h, k-l). In contrast, mice that received the same amount of light stimulation (yoked group), but independent of their behavior, only showed a very low activity-dependent labeling (ca. 4 %, Figure 8 g-h, k-l). As expected, mice receiving no light stimulation had no visible reporter expression.

The results indicate that the repeated pairing between light stimulation and head direction-dependent activity was essential for Cal-Light mediated labeling, and neurons that were inconsistently active during periods of light stimulation were filtered out. In fact, Cal-Light labeling was reported to depend on the number of repetitive stimulations [26], so that a threshold of minimum light stimulations during simultaneous neuronal activity seems likely. Consequently, the resulting coincidence between the individual neuronal activity during light stimulation would not have been high enough to result in a sufficient number of stimulations in the yoked group, as the head direction dependency was not given. Interestingly, a labeling system that would not require multiple stimulations to reach sufficient expression would not be able to filter out unspecific from behavior-specific activity in this way. Thus, using such a system would have resulted in the unspecific labeling of a majority of neurons rather than the low eGFP expression found in yoked mice. However, the number of mice that satisfied the inclusion criteria was too low (n= 5; 2 EXP, 2 Yoke, 1 NoLight) to quantify the resulting labeling of neuronal ensembles more in detail.

5.1.2 From behavior-dependent labeling to causality

Regardless of its potential, behavior-dependent labeling of functional ensembles cannot be the sole solution to investigate causality between behavior and underlying neuronal activity. Instead, it can serve as a starting point to assign neuronal ensembles to correlated behavioral expressions. Selected ensembles can then be probed further to investigate how their activity contributes to behavior. Here, the sophisticated array of tools available in neurophysiology and neuroanatomy will be instrumental [5,8,106].

For example, the neuronal ensembles that were active during specific head direction and subsequently labeled with Cal-Light were not identified as *head direction cells* by conventional means. I.e., their activity was not measured during the task [108,109,112]. However, their head direction specificity was indirectly deduced by the combination of experimental and yoke groups and the fact that the ADN was previously described to contain head direction cells [108]. Here, a direct measurement of head direction tuned activity during the task, e.g., with Ca^{2+} imaging, would clarify ambiguities concerning the label selectivity of the detected behavior and coincidental neuronal activity.

A straightforward solution would be to simultaneously express Ca^{2+} indicators and Cal-Light in the same neurons and measure the activity during behavioral expressions of interest. Here, the identification of active cells during imaging can be compared to the emerging behavior-dependent labels in a two-channel setup. Once labeled, the behavior specificity can be further investigated. If the neuronal ensemble is stable, the same type of behavioral expression should envoke the same neurons under the same conditions – i.e., Cal-Light labeled neurons should be prominently active.

As Cal-Light is a relatively new system, it will be necessary to characterize the exact parameters accompanying behavior-induced, activity-dependent labeling to understand the accuracy and limits of this approach [26,27,29]. Thus, especially in more complex investigations of neuronal activity underlying behavior, it will be beneficial to directly measure the correlation between neuronal activity and expressed behavior with multi-dimensional data – i.e., ethology, connectomics, and physiology.

Finally, to understand a potential causal link between a labeled neuronal ensemble and the investigated behavioral expression, it would be essential to manipulate the neuronal activity and investigate its effect on the behavior [8]. Here, it would be beneficial if the closed-loop solution would detect a behavioral onset as fast as possible – i.e., by predicting the behavioral expression before its onset rather than reacting to it – and then allow the acute manipulation of a selected neuronal ensemble. In this case, a causal relationship would be implied if the disruption of the neuronal ensemble's activity pattern results in a terminated or inhibited behavior. Complementary to this, direct, behavior-dependent optogenetic excitation and inhibition [24,113–115] of neuronal activity is possible using DLStream. However, in the employed setup, the delay between detection and stimulation was ca. 150 ms, which might prove too slow for some experiments but was fast enough to target activity-triggered calcium dynamics with Cal-Light [26,27]. Optimizing the setup might allow faster feedback times as our hardware limited the effective use of the underlying software performance of DLStream.

5.1.3 Performance of DLStream

Regarding the limits of DLStream, it is essential to note that all real-time applications are limited by the system's latency and sample rate. While the latency – i.e., the time until a system reacts to a given input – ideally should be as low as possible but is limited by the computational complexity of the required detection and processing, the required sample rate depends on several factors.

As observers, experimenters often record and interpret an animal's behavior by taking its movement to approximate the underlying intention or state of mind. Building on this generalization, behavior can be defined, categorized, and even sequenced by examining estimations of the animal's movement [51,52,55,116]. However, a researcher might only need the broadest category of movements, or behavioral states, to understand an animal's principle behavior. In contrast, to correlate behavior with neuronal computations, it might be necessary to obtain fast, accurate posture sequences to classify behavioral expressions on a sub-second scale [52,54,106].

The standard temporal resolution (30 Hz) employed in the optogenetic experiment enables behavior-dependent manipulation of a wide range of activities a rodent might perform during a task (see chapter 4.1.7). Swift movements, however, like whisker movement [117,118] and pupil contraction [119,120], might not be fully detected in this configuration. However, by lowering the image resolution and utilizing different network architectures, fast behavioral expressions can be fully captured with DLStream (up to 130 Hz; Table 1). Notably, the limiting factor of DLStream's performance can be traced back to the pose estimation inference time [39–41,43,47]. However, with the rise of real-time, closed-loop toolkits [1,37,38], including DLStream, the providers of open-source pose estimation models have integrated optimized architectures for low inference time [43,46,47]. From a pure performance perspective, the use of faster neural network architectures (e.g., MobilNetV2 [104]) already increases the available framerate by a factor of four (30 to 130 fps, Table 1). This improvement is consistent with the recent large-scale benchmark tests run by DLC [46,47]. Such fast inference times lay the foundation for implementing machine learning-based classification of complex behavior (see chapter 3.3). The use of machine learning increases the range and speed of detection considerably (see chapter 5.1.4 for further details), which will be crucial in disentangling causal relationships between behavioral expressions and coincidental neuronal activity.

5.1.4 Real-time machine learning-based behavior classification in DLStream

Complex behavior analysis tools based on machine learning (ML) classification are actively developed using pose estimation as input [53,54,76,77]. Notably, the usefulness in closed-loop experiments is dependent on the complexity of the behavioral expressions of interest. For example, behavioral expressions that can be described by a few feature changes (e.g., angular changes in head movement) can be calculated without the need for further ML integration. Such behavioral expression can be integrated into DLStream as *triggers* based on single posture or sequential postural information [1] (Supplementary Information 9.3.3). However, complex behavioral expressions (e.g., grooming or social behavior, Supplementary Figure C a-b) would likely require a more sophisticated ML approach to achieve reliable detection [37,38,49,50,52–54,76,77,106].

Fortunately, DLStream was developed as a sustainable, open-source toolkit to facilitate a wide range of experiments across research groups independent of the in-domain knowledge researchers might need to develop their own custom solution. Therefore, the architecture, GUI, and documentation were built such that new users could design their own experiments from the start (see also Supplementary Information 9.3). This includes the publication of several example modules that facilitate the integration of closed-loop experiments (Supplementary Information 9.3.5; Supplementary Table A).

Since the initial publication of DLStream [1], several updates have increased the spectrum of available experimental designs (Supplementary Table A, Supplementary Table B) and pose estimation models. A fundamental development was integrating ML classification of complex behavior based on available open-source solutions [54,76]. An additional set of modules was designed that allows the integration of ML classifiers into DLStream. Fundamentally, ML classification can serve as a foundation to explore novel behavioral patterns and correlated neuronal activity [50,52,54,56]. Specifically, in combination with behavior-dependent labeling (DLStream + Cal-Light), ML classifiers could be used first to identify functional ensembles related to complex behavioral expressions and second to manipulate them selectively to probe causal relationships.

In principle, ML classification based on pose estimation requires three steps to integrate efficiently into closed-loop experiments. First, pose estimation needs to be collected within a classifier-specific time window, and a set of features need to be extracted that match the classifier's specifications. In B-SoiD [54], an unsupervised behavior classification approach, behavioral episodes of 100 ms are captured and analyzed. Briefly, in DLStream, an experiment consists of a behavior-dependent trigger, the dynamic control of stimulation devices, and an experimental protocol that orchestrates the basic structure of the experiment (see chapter 4.1.4 and Supplementary Information 9.3.2). With ML classification, an additional *FeatureExtraction* module collects sequences of pose estimation, extracts the relevant features, and passes them to the *Classifier* module (see Supplementary Information 9.3.1, 9.3.6). The *Classifier* module acts as an interface for the specific classifier type (e.g., RandomForest) and origin (e.g., B-SoiD or SimBA). It provides a consistent way to integrate machine learning classifiers into *trigger* modules. However, because ML classification is a time- and resource-intensive process, the effective integration into DLStream relies on real-time optimization with parallel processing (see Supplementary Information 9.3.6, Supplementary Figure C, Supplementary Table E).

5.1.5 Performance of real-time classification in DLStream

In addition, both feature extraction and classification time can be further optimized by specialization (see Supplementary Information 9.3.6). Non-optimized SiMBA-classifiers reach a computation time of 114.04 ±5.98 ms (n = 1000) per cycle. In contrast, optimizing the classifiers before their implementation for real-time use (Supplementary Table E) reduces the classification time by more than tenfold to 9.44 ± 2.19 ms (n = 1000). In principle, a more compact architecture of the classifier would further reduce the classification time. However, it will cost prediction accuracy – e.g., by reducing the number of decision trees in a random forest classifier. The same principle applies to feature extraction. Depending on the number of features needed for the classification, the computational demand increases substantially (standard SiMBA feature extraction with 14 body parts, 490 features: 235.56 ± 4.72 ms). However, with speed-optimized extraction algorithms, the feature extraction time can be reduced to insubstantial durations (optimized SiMBA feature extraction with 14 bp, 55 features: 0.09 ± 0.69 ms; see Supplementary Information 9.3.6, Supplementary Table E).

In comparison, the standard B-SoiD classification reaches 22.88 ± 4.36 ms (standard feature extraction: 38.25 ± 3.20 ms) without any optimization in DLStream (Supplementary Table E). Notably, SiMBA-based classifiers are binary classifiers that only predict the occurrence of a single behavior, although they are usually used in batteries of multiple classifiers offline [76]. In contrast, B-SoiD classifiers predict multiple behavioral expressions simultaneously [54]. Both classifiers seem promising for future use in behavior-dependent experiments. However, optimization steps will be crucial for the effective use of real-time classification and allow higher framerate, real-time pose estimation without additional delays added by the classification (see Supplementary Figure C).

5.1.6 Reliable multiple animal tracking in DLStream

DLStream was initially developed for single animal experiments using DLC 1.11 (Nature Neuroscience Version, [39]). However, the current version of DLStream [66] can utilize pose estimation models from several toolkits, including the latest DLC version [39,46]. Therefore, models from SLEAP [40,43], DLC-LIVE [46], and DLC [39,48] can be fully integrated into DLStream. Additionally, experimental implementations of models exported by DeepPoseKit [41] (LEAP [40], StackedHourglass [121], StackedDenseNet [41]) as well as multiple animal DLC [45] (maDLC) are available.

Specifically, maDLC and SLEAP allow the pose estimation of multiple animals, which was only possible in edge cases before. In multiple animal tracking, the frequent interactions of individuals cause occlusions, complicating the pose estimation of a complete set of key points. Further, multiple animal pose estimation requires predicting and keeping an animal's identity across frames, which was previously only possible in real-time for individuals with distinctive features (e.g., different fur colors) [39,41,43,76]. For offline solutions, the identity is often referred to with reference to both future and past frames [43,122–124], a solution that is not applicable in real-time settings. However, future developments in both SLEAP [43] and maDLC [45] are supposed to include inbuilt identity tracking that only requires data of past occurrences of the same individual.

In this regard, establishing reliable multiple animal tracking will open up closed-loop experiments dependent on social behavior. Here, the challenge will most likely be the precise definition of social triggers and the design of relevant experiments using closed-loop stimulation. ML behavior classification already enables users of SimBA to analyze social behavior in offline settings [76]. Therefore, the leap to real-time, social behavior-dependent experiments seems imminent. The pure classification speed available with DLStream seems promising (see Supplementary Information 9.3.6). However, the performance and accuracy of such applications need to be carefully evaluated before designing experiments. Here, the main challenge will lie in training personalized, accurate machine learning-based classifiers.

5.1.7 Available open-source, pose estimation-based closed-loop systems

Since the initial development of DLStream and the publication of the preprint [125], the use of pose estimation as a basis for real-time tracking of animals in behavior experiments has become more popular. The original authors of DLC [39,47] released a real-time-optimized version of DLC, DLC-Live [46], in collaboration with Bonsai (bonsai-rx.org) and AutoPilot [126]. Two toolkits that allow users to process data streams from several devices and automatize experiments similar to LabVIEW (National Instruments) [127]. However, an actual neuroscientific experiment using DLC-Live has yet to come. So far, the publication of DLC-Live enabled an easier integration of DLC models into DLStream, and their extensive benchmarks are helpful to establish the setup requirements for new user [46,47]. A similar approach by Forys et al. 2020 [38] used the original DLC toolkit to realize a closed-loop experiment with head-fixed mice. Although their implementation reached a low latency and high framerate [38], the published toolkit lacks the complexity and flexibility to be easily integrated into other labs and experiments.

As a more sophisticated example of the DLC-derived closed-loop systems, EthoLoop [37] specialized in the detection of behavior in naturalistic environments. Using 3D object detection, pose estimation, and a sophisticated array of cameras, EthoLoop allows the tracking and stimulation of freely roaming animals in real-time. Unfortunately, the system is built for large-scale setups and requires specialized hardware to be established. However, its use in 3D environments is unprecedented and will likely facilitate the investigation of primates, birds, and other highly agile animals.

Notably, DLStream could also be upgraded to use 3D posture detection as implemented recently by EthoLoop [37] or DANNCE [42]. The use of multiple camera angles to triangulate the animal's position was already shown for DLC-based pose estimation [37,48]. However, multiple camera streams would increase the computational resource demand and most likely increase the available latency.

5.2 Multicolor light sheet fluorescence expansion microscopy

With real-time, closed-loop experiments around the corner, the focus will shift to the biomolecular tools available to capture, manipulate, and measure the complex relationships between behavioral expressions and neuronal activity. Especially concerning the investigation of functional ensembles, the unambiguous characterization of captured neurons will become a central challenge. Here, the disadvantage of classic labeling strategies – i.e., single-color labeling – lies in identifying individual neurons, especially in dense areas (see chapter 3.6). However, it will be crucial to characterize neurons in great detail across large scales to sufficiently understand functional ensembles of behavior. Here, techniques that expand on conventional labeling strategies combined with fast, high-resolution imaging serve as an excellent way to disentangle the underlying complexity.

Therefore, in close collaboration with the Institute for Theoretical and Physical Chemistry at the University of Bonn, a study was conducted to integrate multicolor-based segmentation within the previously established virtual super-resolution LSFEM workflow [57,128]. My main focus was establishing strategies for effective Tetbow-based labeling for tissue expansion and automatic hue-based segmentation (see chapters 3.5 and 3.6). The study's initial focus was to investigate the relationships between CA3 pyramidal neurons and their connected DG counterparts. As the hippocampus consists of multiple densely populated layers and connections are both short and long-range, it provides a well-studied basis for establishing this advanced connectomic analysis. However, the technique's potential in the context of behavior-dependent labeling is remarkable (see chapter 5.3.1).

Notably, the algorithm and its quantitative results are published separately [105] and were not part of this thesis.

5.2.1 Hue-based analysis of expanded tissue

In principle, the power of high contrast, multicolor labeling using Tetbow allows the distinction of closely neighboring neurons in densely layered regions such as the hippocampal formation [62]. Additionally, combining Tetbow with powerful imaging techniques such as LSFEM paves the way for scale-bridging connectomics. However, so far, the system was not demonstrated to work with tissue expansion protocols and virtual super-resolution imaging.

The Tetbow system requires careful evaluation of expression ratios between all viral components for practical use. Notably, the XFP expression needs to be fine-tuned to compensate for any fluorescent loss during tissue expansion (see Supplementary Table D), even if it is minimal [57,58], mainly because the retained fluorescence can be different across XFPs [129]. Interestingly, during experiments, it could not be confirmed that Tetbow's expression levels increase with decreasing tTA vector ratios but rather the other way around [62] (see Supplementary Table D). This stands in contrast to initial reports by the original authors [62] but makes sense considering the expression system (see chapter 3.5).

With optimized expression ratios, Tetbow-based LSFEM will allow the classification of a large number of neurons on an individual level across large-scale tissues with remarkable detail (Figure 11 i). For this, the developed algorithm utilizes both spatial- and color-derived information to segment individual neurons [105] (Figure 11 j) and trace them across imaging volumes.

The immediate advantage lies in the broad-range classification of individual neurons identified and segmented by their unique hue using the inherent large-scale capabilities of LSFEM. Thus, individual neurons can be characterized by their morphology [130–132] and their connectivity investigated with short-to-long-range tracing in dense regions automatically (Figure 11 h). Notably, the possibility to identify large-scale structures with a simultaneous resolution to study spike-like protrusions (Figure 11 f-g) or even synapses [57] will be most relevant in the investigation of large-scale effects by small-scale morphology changes. For example, in numerous brain disorders associated with abnormal dendritic spines [59–61].

5.3 Future directions

The roadmap to investigate behavior-relevant circuits based on the tools at hand seems straightforward. First, the behavior-dependent labeling and consecutive manipulation of neuronal ensembles underlying behavior can be achieved by using closed-loop feedback systems (e.g., DLStream) in combination with biomolecular tools such as Cal-Light. For connectomic analysis, the captured ensembles are then imaged with high resolution at a large scale. Here, techniques such as LSFEM are preferable as they bridge both meso- and nanoscopic scales at fast imaging speed. The imaging data can then be integrated into automatic cell counting and segmentation pipelines that characterize the functional ensembles. Second, the causal relationships between captured ensembles and correlated behavioral expressions can be further probed with closed-loop optogenetic experiments – i.e., by utilizing the Cal-Light system with exchanged reporter proteins. However, the intricate relationships between behavioral expressions and neuronal activity raise several requirements that need to be carefully investigated and optimized along the way.

5.3.1 Limitations of Cal-Light in capturing behavioral expressions

For instance, given the need to stimulate on multiple occasions, a disadvantage of a behavior-dependent labeling approach with the Cal-Light system is the necessity to select behavioral expressions performed *regularly* by the animal. Rare behavioral expressions that cannot be induced during experimental sessions – i.e., a behavior only expressed once every session and that cannot be evoked by reward or paradigm changes - will most likely not satisfy the stimulation threshold. Here, Cal-Light's transient expression of reporter genes comes into play. Spreading the same number of stimulations out over an extended period will most likely not be effective because the low expression level per event and the transient lifetime of reporter proteins will limit Cal-Lights effectiveness, even if some sessions rise above the stimulation threshold. However, a more sensitive or more label-efficient approach might be a viable solution to capture rare behavioral expressions.

The Supernova system [133], for example, uses a tTA/TRE-Cre/loxP enhancing system that boosts low levels of tTA/TRE activation by a Cre/loxP-controlled feedback loop. First, tTA-mediated expression of TRE-Cre results in Cre-recombinase expression. Second, the Cre-mediated recombination of CAG-loxP-stop-loxP-XFP-tTA results in the expression of XFP-tTA, initiating a feedback loop enhancing XFP expression for high molecular contrast. Thus, a supernova-like Cal-Light adaptation would only need very few behavior-dependent light stimulations to activate the enhanced expression system and constantly express the reporter afterward.

In theory, such a system could be generated by combining Cal-Light [26] and Supernova [133] components directly. Cal-Light utilizes the activity-dependent, light-induced release of tTA to express TRE-controlled reporter genes. Thus, rather than driving XFP expression directly, Cal-Light could be used to trigger the TRE-Cre expression and subsequent feedback loop of Supernova [133]. The main advantage being the longer lifetime of the behavior-dependent label and the shorter stimulation requirements. A system like this would be especially beneficial to express functionally active reporters utilized afterward – e.g., ChR/NpHR for optogenetic manipulation or calcium indicators for Ca^{2+} imaging of behavior-dependently labeled neurons. However, a supernova-like Cal-Light approach would require careful verification of the causal relationship between neuronal activity and behavior-dependent labeling due to its increased sensitivity.

5.3.2 Limitations of Cal-Light in connectomic analysis

Another challenge concerns the highly detailed connectomic analysis of captured ensembles. As previously mentioned, the single-color labeling of large quantities of closely neighboring neurons restricts the level of connectomic analysis considerably. Therefore, multicolor, activity-dependent labeling would facilitate a holistic segmentation of individual neurons that form functional ensembles. However, the current version of Cal-Light can only express single XFPs.

A potential solution could be realized by combining the light-induced, activity-dependent release of tTA using Cal-Light [26] (see chapter 3.1) with the tTA/TRE-dependent expression of Tetbow directly. However, it is questionable whether the short-term expression by Cal-Light's tTA release is enough to drive the expression levels required to resolve Tetbow into a wide range of different color hues. Here, an enhancement system or a system that is activated only once would likely be more promising. The original approach, Brainbow [63,64], is Cre-dependent, meaning that the activation of reporter gene expression is initially required but is continuous from that point on. However, it lacks an additional enhancement strategy and was reported to result in low contrast expression [62], which is one of the main reasons Tetbow was developed as its successor.

In contrast, a system enhancing expression levels after initial activation with Cal-Light would facilitate the expression of Tetbow-like color diversity. Here, the Supernova [133] approach comes into mind again. However, the sheer amount of different viral vectors necessary to allow a fully functional version of a Cal-Light/Tetbow/Supernova approach reduces the likelihood of a successful implementation. First, the high viral load could be toxic to neurons. Second, the likelihood of simultaneous effective infection with several vectors is reduced with increasing numbers of components.

A step in the right direction might be scFLARE [29]. The FLARE system works very similar to Cal-Light – i.e., it allows the light-induced, activity-dependent labeling of neurons and utilizes the tTA/TRE system [28]. However, FLARE suffers from the same multi-vector requirement that Cal-Light has. With scFLARE, the original authors generated a derivative of FLARE [28] that reduces the number of vectors necessary to a single vector. Interestingly, the new version was also reported to have greater dynamic range and robustness than the original version [29].

In this regard, the rise of multi-color ensemble labeling consequently will result in an increased need for the integration of automatic quantification to tackle the highly complex imaging data. Here, developments like the hue-based segmentation algorithm [105] and AI-based whole brain cell counting [134] will be crucial in analyzing captured ensembles' projections, morphologies, and local architectures.

5.3.3 Improvements to DLStream behavior detection

High-throughput connectomic analysis aside, the next step to disentangle the causal relationship between complex behavioral expressions and neuronal activity relies on the fast, robust identification of behavioral expressions. Here, the complete integration of fast, reliable real-time behavior classification into closed-loop experiments will be essential. Current offline solutions use two distinctly different approaches to tackle an animal's behavioral repertoire during any observed session. However, in principle, independent of the solution, a machine learning algorithm – i.e., a classifier – is trained to classify data into previously learned categories.

Supervised classification [76] directly takes input from user-defined annotations and trains a classifier to detect the previously defined behavioral expressions based on features extracted from pose estimation (see Supplementary Figure C a-b). The main advantage of this is the direct control a user has over the initial definition of the behavioral expression. This you-get-what-you-label approach is advantageous if researchers are only interested in a limited number of behavioral expressions. However, fully annotating training sets for each behavior of interest is a time-consuming disadvantage that results in rigid categorical definitions that can be prone to inherent biases and rater fatigue. Additionally, this approach is not easily scalable, especially in more generalized investigations to untangle the complete behavioral repertoire of animals. Inherently, a supervised algorithm is unable to give new insights into previously unknown structures.

In contrast, unsupervised classification [52–56,77] limits the researcher's influence on the definition of hyper-parameters and includes an additional analytical stage that reveals underlying structures in the observed behavior. Existing solutions use different approaches to expose predominant structures. However, independent of the particular method, the resulting data can be used to train classifiers similar to the supervised approach directly. While unsupervised behavior analysis can reveal behavioral structures in a previously unmatched level of detail, the vast amount of extracted data needs to be heavily curated to align identified structures to preexisting behavioral stereotypes.

For solutions like DLStream, the key advantage will lie in the collaboration with and integration of open-source toolkits such as BSoID [54], SimBA [76], and others [52,53,56,77,106]. For example, the DLStream integration of SimBA-based classification, a fully supervised classification tool, combined with multiple animal tracking, showed promising results in recent internal tests (see Supplementary Information 9.3.6). At the same time, *pure-predict* [135] optimized classification speed seems to be particularly well suited for real-time requirements (see Methods 7.13; Supplementary Table E) and will facilitate the integration of solutions that are not real-time optimized by design. In addition, ML analysis yields the prospect of predicting behavior, for example, by matching initial elements of a uniquely arranged behavioral sequence, further reducing the latency to react to a behavioral expression. *Predicting* behavioral expressions will be especially relevant to investigate the causal relationship of neuronal computations that evoke behavioral expressions. Here, the main disadvantage of current behavior detection strategies is the *reactive* detection based on complete expressions – i.e., the behavioral expression must be observed in full before it is recognized. Any underlying neuronal activity partially preceding the behavioral expression or happening only during the very beginning will likely not be captured using this approach. Therefore, the future direction will be the generation of *predictive* detection strategies that can identify behavioral expressions based on incomplete expressions – i.e., detecting them before they are completed. In this regard, the combination of pose estimation with alternative, non-video-based tracking (e.g., eye-tracking; [136]) or additional behavioral dimensions such as vocalization [137–139] might lead to a solution for researchers interested in capturing truly holistic behavioral data.

5.4 Conclusion

The combination of DLStream and Cal-Light allows to capture neuronal ensembles active during selected, unconstrained behavioral expressions in real-time. In this thesis, DLStream specifically was used to optogenetically label active neurons during selected episodes head direction in mice. Unlike previous implementations of Cal-Light-like systems [26,28], the light induction with DLStream is not dependent on the interaction with a lever or other device but captures unconstrained behavioral expressions in a freely moving setup. As such, DLStream elevates the potential use of light-induced, activity-dependent labeling systems to unrestricted behavior-dependency. Notably, the toolkit has no apparent limitations for the use with different organisms and other experimental paradigms [1]. Its capabilities can be directly translated to a wide range of organisms utilizing the generalizability of pose estimation models across different species [41,43,48].

Additionally, detailed documentation and several example modules were published to further facilitate its use along with DLStream as an open-source software package [66].

I believe that the use of DLStream will facilitate the characterization of causal relationships between behavioral expressions and underlying neuronal activity, especially if activity-dependent labeling can be combined with multicolor approaches like Tetbow.

6 References

1. Schweihoff, J. F. *et al.* DeepLabStream enables closed-loop behavioral experiments using deep learning-based markerless, real-time posture detection. *Communications biology* **4,** 130; 10.1038/s42003-021-01654-9 (2021).
2. Abbott, L. F. *et al.* The Mind of a Mouse. *Cell* **182,** 1372–1376; 10.1016/j.cell.2020.08.010 (2020).
3. Berman, G. J. Measuring behavior across scales. *BMC biology* **16,** 23; 10.1186/s12915-018-0494-7 (2018).
4. Anderson, D. J. & Perona, P. Toward a science of computational ethology. *Neuron* **84,** 18–31; 10.1016/j.neuron.2014.09.005 (2014).
5. Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A. & Poeppel, D. Neuroscience Needs Behavior: Correcting a Reductionist Bias. *Neuron* **93,** 480–490; 10.1016/j.neuron.2016.12.041 (2017).
6. Gomez-Marin, A., Paton, J. J., Kampff, A. R., Costa, R. M. & Mainen, Z. F. Big behavioral data: psychology, ethology and the foundations of neuroscience. *Nature Neuroscience* **17,** 1455 EP -; 10.1038/nn.3812 (2014).
7. Silvanto, J. & Pascual-Leone, A. Why the assessment of causality in brain-behavior relations requires brain stimulation. *Journal of cognitive neuroscience* **24,** 775–777; 10.1162/jocn_a_00193 (2012).
8. Wolff, S. B. & Ölveczky, B. P. The promise and perils of causal circuit manipulations. *Current opinion in neurobiology* **49,** 84–94; 10.1016/j.conb.2018.01.004 (2018).
9. Otchy, T. M. *et al.* Acute off-target effects of neural circuit manipulations. *Nature* **528,** 358–363; 10.1038/nature16442 (2015).
10. Boyden, E. S. Optogenetics and the future of neuroscience. *Nature Neuroscience* **18,** 1200–1201; 10.1038/nn.4094 (2015).
11. Rajasethupathy, P., Ferenczi, E. & Deisseroth, K. Targeting Neural Circuits. *Cell* **165,** 524–534; 10.1016/j.cell.2016.03.047 (2016).
12. Deisseroth, K. Optogenetics: 10 years of microbial opsins in neuroscience. *Nature Neuroscience* **18,** 1213–1225; 10.1038/nn.4091 (2015).
13. Kwon, J.-T. *et al.* Optogenetic activation of presynaptic inputs in lateral amygdala forms associative fear memory. *Learning & memory (Cold Spring Harbor, N.Y.)* **21,** 627–633; 10.1101/lm.035816.114 (2014).
14. Sousa, A. F. de *et al.* Optogenetic reactivation of memory ensembles in the retrosplenial cortex induces systems consolidation. *Proceedings of the National Academy of Sciences of the United States of America* **116,** 8576–8581; 10.1073/pnas.1818432116 (2019).
15. Oishi, N. *et al.* Artificial association of memory events by optogenetic stimulation of hippocampal CA3 cell ensembles. *Molecular Brain* **12,** 2; 10.1186/s13041-018-0424-1 (2019).
16. Marshel, J. H. *et al.* Cortical layer-specific critical dynamics triggering perception. *Science (New York, N.Y.)* **365**; 10.1126/science.aaw5202 (2019).
17. Carrillo-Reid, L., Han, S., Yang, W., Akrouh, A. & Yuste, R. Controlling Visually Guided Behavior by Holographic Recalling of Cortical Ensembles. *Cell* **178,** 447-457.e5; 10.1016/j.cell.2019.05.045 (2019).

18. Magno, L. A. V. *et al.* Optogenetic Stimulation of the M2 Cortex Reverts Motor Dysfunction in a Mouse Model of Parkinson's Disease. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **39,** 3234–3248; 10.1523/JNEUROSCI.2277-18.2019 (2019).

19. Ebina, T. *et al.* Arm movements induced by noninvasive optogenetic stimulation of the motor cortex in the common marmoset. *Proceedings of the National Academy of Sciences of the United States of America*; 10.1073/pnas.1903445116 (2019).

20. Baumgartner, C., Koren, J. P. & Rothmayer, M. Automatic Computer-Based Detection of Epileptic Seizures. *Frontiers in neurology* **9,** 639; 10.3389/fneur.2018.00639 (2018).

21. Krook-Magnuson, E., Armstrong, C., Oijala, M. & Soltesz, I. On-demand optogenetic control of spontaneous seizures in temporal lobe epilepsy. *Nature communications* **4,** 1376; 10.1038/ncomms2376 (2013).

22. Paz, J. T. *et al.* Closed-loop optogenetic control of thalamus as a tool for interrupting seizures after cortical injury. *Nature Neuroscience* **16,** 64–70; 10.1038/nn.3269 (2013).

23. Chen, R. *et al.* Deep brain optogenetics without intracranial surgery. *Nature biotechnology* **39,** 161–164; 10.1038/s41587-020-0679-9 (2021).

24. Boyden, E. S., Zhang, F., Bamberg, E., Nagel, G. & Deisseroth, K. Millisecond-timescale, genetically targeted optical control of neural activity. *Nature Neuroscience* **8,** 1263–1268; 10.1038/nn1525 (2005).

25. Grosenick, L., Marshel, J. H. & Deisseroth, K. Closed-loop and activity-guided optogenetic control. *Neuron* **86,** 106–139; 10.1016/j.neuron.2015.03.034 (2015).

26. Lee, D., Hyun, J. H., Jung, K., Hannan, P. & Kwon, H.-B. A calcium- and light-gated switch to induce gene expression in activated neurons. *Nature biotechnology* **35,** 858 EP -; 10.1038/nbt.3902 (2017).

27. Ebner, C. *et al.* Optically Induced Calcium-Dependent Gene Activation and Labeling of Active Neurons Using CaMPARI and Cal-Light. *Frontiers in synaptic neuroscience* **11,** 16; 10.3389/fnsyn.2019.00016 (2019).

28. Wang, W. *et al.* A light- and calcium-gated transcription factor for imaging and manipulating activated neurons. *Nature biotechnology* **35,** 864 EP -; 10.1038/nbt.3909 (2017).

29. Sanchez, M. I., Nguyen, Q.-A., Wang, W., Soltesz, I. & Ting, A. Y. Transcriptional readout of neuronal activity via an engineered Ca2+-activated protease. *Proceedings of the National Academy of Sciences of the United States of America*; 10.1073/pnas.2006521117 (2020).

30. Paulk, A. C., Kirszenblat, L., Zhou, Y. & van Swinderen, B. Closed-Loop Behavioral Control Increases Coherence in the Fly Brain. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **35,** 10304–10315; 10.1523/JNEUROSCI.0691-15.2015 (2015).

31. Solari, N., Sviatkó, K., Laszlovszky, T., Hegedüs, P. & Hangya, B. Open Source Tools for Temporally Controlled Rodent Behavior Suitable for Electrophysiology and Optogenetic Manipulations. *Frontiers in systems neuroscience* **12,** 18; 10.3389/fnsys.2018.00018 (2018).

32. Thurley, K. & Ayaz, A. Virtual reality systems for rodents. *Current zoology* **63,** 109–119; 10.1093/cz/zow070 (2017).

33. Bourboulou, R. *et al.* Dynamic control of hippocampal spatial coding resolution by local visual cues. *eLife* **8**; 10.7554/eLife.44487 (2019).

34. Fuhrmann, F. *et al.* Locomotion, Theta Oscillations, and the Speed-Correlated Firing of Hippocampal Neurons Are Controlled by a Medial Septal Glutamatergic Circuit. *Neuron* **86,** 1253–1264; 10.1016/j.neuron.2015.05.001 (2015).

35. Musso, P.-Y. *et al.* Closed-loop optogenetic activation of peripheral or central neurons modulates feeding in freely moving Drosophila. *eLife* **8**; 10.7554/eLife.45636 (2019).

36. Štih, V., Petrucco, L., Kist, A. M. & Portugues, R. Stytra: An open-source, integrated system for stimulation, tracking and closed-loop behavioral experiments. *PLoS computational biology* **15,** e1006699; 10.1371/journal.pcbi.1006699 (2019).

37. Nourizonoz, A. *et al.* EthoLoop: automated closed-loop neuroethology in naturalistic environments. *Nature methods* **17,** 1052–1059; 10.1038/s41592-020-0961-2 (2020).

38. Forys, B. J., Xiao, D., Gupta, P. & Murphy, T. H. Real-Time Selective Markerless Tracking of Forepaws of Head Fixed Mice Using Deep Neural Networks. *eNeuro* **7**; 10.1523/ENEURO.0096-20.2020 (2020).

39. Mathis, A. *et al.* DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience* **21,** 1281–1289; 10.1038/s41593-018-0209-y (2018).

40. Pereira, T. D. *et al.* Fast animal pose estimation using deep neural networks. *Nature methods* **16,** 117–125; 10.1038/s41592-018-0234-5 (2019).

41. Graving, J. M. *et al.* DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. *eLife* **8,** e47994; 10.7554/eLife.47994 (2019).

42. Dunn, T. W. *et al.* Geometric deep learning enables 3D kinematic profiling across species and environments. *Nature methods*; 10.1038/s41592-021-01106-6 (2021).

43. Pereira, T. D. *et al.* SLEAP: Multi-animal pose tracking. *BioRxiv*; 10.1101/2020.08.31.276246 (2020).

44. Hebert, L., Ahamed, T., Costa, A. C., O'Shaughnessy, L. & Stephens, G. J. WormPose: Image synthesis and convolutional networks for pose estimation in C. elegans. *PLoS computational biology* **17,** e1008914; 10.1371/journal.pcbi.1008914 (2021).

45. Lauer, J. *et al.* Multi-animal pose estimation and tracking with DeepLabCut. *BioRxiv*; 10.1101/2021.04.30.442096 (2021).

46. Kane, G. A., Lopes, G., Saunders, J. L., Mathis, A. & Mathis, M. W. Real-time, low-latency closed-loop feedback using markerless posture tracking. *eLife* **9**; 10.7554/eLife.61909 (2020).

47. Mathis, A. & Warren, R. On the inference speed and video-compression robustness of DeepLabCut. *BioRxiv*; 10.1101/457242 (2018).

48. Nath, T. *et al.* Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature Protocols* **14,** 2152–2176; 10.1038/s41596-019-0176-0 (2019).

49. Sturman, O. *et al.* Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. *Neuropsychopharmacology : official publication of the American College of Neuropsychopharmacology* **45,** 1942–1952; 10.1038/s41386-020-0776-y (2020).

50. Markowitz, J. E. *et al.* The Striatum Organizes 3D Behavior via Moment-to-Moment Action Selection. *Cell* **174,** 44-58.e17; 10.1016/j.cell.2018.04.019 (2018).

51. Wang, Z., Mirbozorgi, S. A. & Ghovanloo, M. An automated behavior analysis system for freely moving rodents using depth image. *Medical & biological engineering & computing* **56,** 1807–1821; 10.1007/s11517-018-1816-1 (2018).

52. Wiltschko, A. B. *et al.* Mapping Sub-Second Structure in Mouse Behavior. *Neuron* **88,** 1121–1135; 10.1016/j.neuron.2015.11.031 (2015).

53. Luxem, K., Fuhrmann, F., Kürsch, J., Remy, S. & Bauer, P. Identifying Behavioral Structure from Deep Variational Embeddings of Animal Motion. *BioRxiv*; 10.1101/2020.05.14.095430 (2020).

54. Hsu, A. I. & Yttri, E. A. B-SOiD, an open-source unsupervised algorithm for identification and fast prediction of behaviors. *Nature communications* **12,** 5188; 10.1038/s41467-021-25420-x (2021).

55. Berman, G. J., Choi, D. M., Bialek, W. & Shaevitz, J. W. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of the Royal Society, Interface* **11**; 10.1098/rsif.2014.0672 (2014).

56. Klibaite, U., Berman, G. J., Cande, J., Stern, D. L. & Shaevitz, J. W. An unsupervised method for quantifying the behavior of paired animals. *Physical biology* **14,** 15006; 10.1088/1478-3975/aa5c50 (2017).

57. Bürgers, J. *et al.* Light-sheet fluorescence expansion microscopy: fast mapping of neural circuits at super resolution. *Neurophotonics* **6,** 15005; 10.1117/1.NPh.6.1.015005 (2019).

58. Gao, R. *et al.* Cortical column and whole-brain imaging with molecular contrast and nanoscale resolution. *Science (New York, N.Y.)* **363**; 10.1126/science.aau8302 (2019).

59. Benson, C. A. *et al.* Dendritic Spine Dynamics after Peripheral Nerve Injury: An Intravital Structural Study. *The Journal of neuroscience : the official journal of the*

Society for Neuroscience **40,** 4297–4308; 10.1523/JNEUROSCI.2858-19.2020 (2020).

60. Krueppel, R., Remy, S. & Beck, H. Dendritic integration in hippocampal dentate granule cells. *Neuron* **71,** 512–528; 10.1016/j.neuron.2011.05.043 (2011).

61. Penzes, P., Cahill, M. E., Jones, K. A., VanLeeuwen, J.-E. & Woolfrey, K. M. Dendritic spine pathology in neuropsychiatric disorders. *Nature Neuroscience* **14,** 285–293; 10.1038/nn.2741 (2011).

62. Sakaguchi, R., Leiwe, M. N. & Imai, T. Bright multicolor labeling of neuronal circuits with fluorescent proteins and chemical tags. *eLife* **7**; 10.7554/eLife.40350 (2018).

63. Weissman, T. A. & Pan, Y. A. Brainbow: new resources and emerging biological applications for multicolor genetic labeling and analysis. *Genetics* **199,** 293–306; 10.1534/genetics.114.172510 (2015).

64. Cai, D., Cohen, K. B., Luo, T., Lichtman, J. W. & Sanes, J. R. Improved tools for the Brainbow toolbox. *Nature methods* **10,** 540–547; 10.1038/nmeth.2450 (2013).

65. Livet, J. *et al.* Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. *Nature* **450,** 56–62; 10.1038/nature06293 (2007).

66. Schweihoff, J., Matvey Loshakov & Schwarz Lab. *SchwarzNeuroconLab/DeepLabStream v1.4* (Zenodo, 2021).

67. O'Keefe, J. Place units in the hippocampus of the freely moving rat. *Experimental Neurology* **51,** 78–109; 10.1016/0014-4886(76)90055-8 (1976).

68. Abdelfattah, A. S. *et al.* Bright and photostable chemigenetic indicators for extended in vivo voltage imaging. *Science (New York, N.Y.)* **365,** 699–704; 10.1126/science.aav6416 (2019).

69. Skocek, O. *et al.* High-speed volumetric imaging of neuronal activity in freely moving rodents. *Nature methods* **15,** 429–432; 10.1038/s41592-018-0008-0 (2018).

70. Ghosh, K. K. *et al.* Miniaturized integration of a fluorescence microscope. *Nature methods* **8,** 871–878; 10.1038/nmeth.1694 (2011).

71. Szabo, V., Ventalon, C., Sars, V. de, Bradley, J. & Emiliani, V. Spatially selective holographic photoactivation and functional fluorescence imaging in freely behaving mice with a fiberscope. *Neuron* **84,** 1157–1169; 10.1016/j.neuron.2014.11.005 (2014).

72. Winter, D. A. *Biomechanics and motor control of human movement.* 4th ed. (Wiley; Chichester : John Wiley [distributor], Hoboken, N.J., 2009).

73. Vargas-Irwin, C. E. *et al.* Decoding complete reach and grasp actions from local primary motor cortex populations. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **30,** 9659–9669; 10.1523/JNEUROSCI.5443-09.2010 (2010).

74. Maghsoudi, O. H., Tabrizi, A. V., Robertson, B. & Spence, A. Superpixels based marker tracking vs. hue thresholding in rodent biomechanics application. In *2017 51st Asilomar Conference on Signals, Systems, and Computers* (IEEE102017), pp. 209–213.

75. Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M. & Schiele, B. DeeperCut: A Deeper, Stronger, and Faster Multi-person Pose Estimation Model. In *Computer Vision – ECCV 2016,* edited by B. Leibe, J. Matas, N. Sebe & M. Welling (Springer International Publishing, Cham, 2016), Vol. 9910, pp. 34–50.

76. Nilsson, S. R. O. *et al.* Simple Behavioral Analysis (SimBA) – an open source toolkit for computer classification of complex social behaviors in experimental animals. *BioRxiv*; 10.1101/2020.04.19.049452 (2020).

77. Graving, J. M. & Couzin, I. D. VAE-SNE: a deep generative model for simultaneous dimensionality reduction and clustering. *BioRxiv*; 10.1101/2020.07.17.207993 (2020).

78. Chang, J.-B. *et al.* Iterative expansion microscopy. *Nat Methods* **14,** 593–599; 10.1038/nmeth.4261 (2017).

79. Chen, F., Tillberg, P. W. & Boyden, E. S. Optical imaging. Expansion microscopy. *Science (New York, N.Y.)* **347,** 543–548; 10.1126/science.1260088 (2015).

80. Chozinski, T. J. *et al.* Expansion microscopy with conventional antibodies and fluorescent proteins. *Nature methods* **13,** 485–488; 10.1038/nmeth.3833 (2016).

81. Zhu, D., Larin, K. V., Luo, Q. & Tuchin, V. V. Recent progress in tissue optical clearing. *Laser & photonics reviews* **7,** 732–757; 10.1002/lpor.201200056 (2013).

82. Vigouroux, R. J., Belle, M. & Chédotal, A. Neuroscience in the third dimension: shedding new light on the brain with tissue clearing. *Molecular Brain* **10,** 33; 10.1186/s13041-017-0314-y (2017).

83. Tainaka, K., Kuno, A., Kubota, S. I., Murakami, T. & Ueda, H. R. Chemical Principles in Tissue Clearing and Staining Protocols for Whole-Body Cell Profiling. *Annual review of cell and developmental biology* **32,** 713–741; 10.1146/annurev-cellbio-111315-125001 (2016).

84. Richardson, D. S. & Lichtman, J. W. Clarifying Tissue Clearing. *Cell* **162,** 246–257; 10.1016/j.cell.2015.06.067 (2015).

85. Schwarz, M. K. *et al.* Fluorescent-protein stabilization and high-resolution imaging of cleared, intact mouse brains. *PloS one* **10,** e0124650; 10.1371/journal.pone.0124650 (2015).

86. Voie, A. H., Burns, D. H. & Spelman, F. A. Orthogonal-plane fluorescence optical sectioning: three-dimensional imaging of macroscopic biological specimens. *Journal of microscopy* **170,** 229–236; 10.1111/j.1365-2818.1993.tb03346.x (1993).

87. Silvestri, L., Bria, A., Sacconi, L., Iannello, G. & Pavone, F. S. Confocal light sheet microscopy: micron-scale neuroanatomy of the entire mouse brain. *Optics express* **20,** 20582–20598; 10.1364/OE.20.020582 (2012).

88. Baumgart, E. & Kubitscheck, U. Scanned light sheet microscopy with confocal slit detection. *Optics express* **20,** 21805–21814; 10.1364/OE.20.021805 (2012).

89. Doerr, J. *et al.* Whole-brain 3D mapping of human neural transplant innervation. *Nature communications* **8,** 14162; 10.1038/ncomms14162 (2017).

90. Niedworok, C. J. *et al.* Charting monosynaptic connectivity maps by two-color light-sheet fluorescence microscopy. *Cell reports* **2,** 1375–1386; 10.1016/j.celrep.2012.10.008 (2012).

91. Dodt, H.-U. *et al.* Ultramicroscopy: three-dimensional visualization of neuronal networks in the whole mouse brain. *Nature methods* **4,** 331–336; 10.1038/nmeth1036 (2007).

92. Jefferis, G. S. X. E. & Livet, J. Sparse and combinatorial neuron labelling. *Current opinion in neurobiology* **22,** 101–110; 10.1016/j.conb.2011.09.010 (2012).

93. Pasternak, J. F. & Woolsey, T. A. On the "selectivity" of the Golgi-Cox method. *The Journal of comparative neurology* **160,** 307–312; 10.1002/cne.901600304 (1975).

94. Valverde, F. The Golgi Method. A Tool for Comparative Structural Analyses. In *Contemporary Research Methods in Neuroanatomy,* edited by W. J. H. Nauta & S. O. E. Ebbesson (Springer Berlin Heidelberg, Berlin, Heidelberg, 1970), pp. 12–31.

95. Lee, G. & Saito, I. Role of nucleotide sequences of loxP spacer region in Cre-mediated recombination. *Gene* **216,** 55–65; 10.1016/S0378-1119(98)00325-4 (1998).

96. Stark, W. M., Boocock, M. R. & Sherratt, D. J. Catalysis by site-specific recombinases. *Trends in genetics : TIG* **8,** 432–439 (1992).

97. Branda, C. S. & Dymecki, S. M. Talking about a Revolution. *Developmental Cell* **6,** 7–28; 10.1016/S1534-5807(03)00399-X (2004).

98. Deuschle, U., Meyer, W. K. & Thiesen, H. J. Tetracycline-reversible silencing of eukaryotic promoters. *Molecular and cellular biology* **15,** 1907–1914; 10.1128/mcb.15.4.1907 (1995).

99. Gossen, M. & Bujard, H. Tight control of gene expression in mammalian cells by tetracycline-responsive promoters. *Proceedings of the National Academy of Sciences of the United States of America* **89,** 5547–5551 (1992).

100. Hillen, W. & Berens, C. Mechanisms underlying expression of Tn10 encoded tetracycline resistance. *Annual review of microbiology* **48,** 345–369; 10.1146/annurev.mi.48.100194.002021 (1994).

101. Kobiler, O., Lipman, Y., Therkelsen, K., Daubechies, I. & Enquist, L. W. Herpesviruses carrying a Brainbow cassette reveal replication and expression of limited numbers of incoming genomes. *Nature communications* **1,** 146; 10.1038/ncomms1145 (2010).

102. Cetin, A., Komai, S., Eliava, M., Seeburg, P. H. & Osten, P. Stereotaxic gene delivery in the rodent brain. *Nature Protocols* **1,** 3166–3173; 10.1038/nprot.2006.450 (2006).

103. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEEMonday, June 27, 2016 - Thursday, June 30, 2016), pp. 770–778.

104. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *2018 IEEE/CVF Conference on Computer 2018,* pp. 4510–4520.

105. Heysel, J. L. Abbildung und farbtonbasierte Analyse ausgedehnter neuronaler Strukturen. Dissertation. Rheinische Friedrich-Wilhelms-Universität Bonn, 2021.

106. Datta, S. R., Anderson, D. J., Branson, K., Perona, P. & Leifer, A. Computational Neuroethology: A Call to Action. *Neuron* **104,** 11–24; 10.1016/j.neuron.2019.09.038 (2019).

107. Fosque, B. F. *et al.* Neural circuits. Labeling of active neural circuits in vivo with designed calcium integrators. *Science (New York, N.Y.)* **347,** 755–760; 10.1126/science.1260922 (2015).

108. Taube, J. S. Head direction cells recorded in the anterior thalamic nuclei of freely moving rats. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **15,** 70–86; 10.1523/JNEUROSCI.15-01-00070.1995 (1995).

109. Taube, J. S., Muller, R. U. & Ranck, J. B. Head-direction cells recorded from the postsubiculum in freely moving rats. II. Effects of environmental manipulations. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **10,** 436–447 (1990).

110. Yoder, R. M. & Taube, J. S. Head Direction Cell Activity in Mice: Robust Directional Signal Depends on Intact Otolith Organs. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **29,** 1061–1076; 10.1523/JNEUROSCI.1679-08.2009 (2009).

111. Valerio, S. & Taube, J. S. Head Direction Cell Activity Is Absent in Mice without the Horizontal Semicircular Canals. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **36,** 741–754; 10.1523/JNEUROSCI.3790-14.2016 (2016).

112. Sharp, P. E. Neural Representations of Direction (Head Direction Cells). In *Encyclopedia of Behavioral Neuroscience* (Elsevier2010), pp. 348–355.

113. Josselyn, S. A. The past, present and future of light-gated ion channels and optogenetics. *eLife* **7**; 10.7554/eLife.42367 (2018).

114. Nagel, G. *et al.* Channelrhodopsin-1: a light-gated proton channel in green algae. *Science (New York, N.Y.)* **296,** 2395–2398; 10.1126/science.1072068 (2002).

115. Han, X. & Boyden, E. S. Multiple-color optical activation, silencing, and desynchronization of neural activity, with single-spike temporal resolution. *PloS one* **2,** e299; 10.1371/journal.pone.0000299 (2007).

116. Stacher Hörndli, C. N. *et al.* Complex Economic Behavior Patterns Are Constructed from Finite, Genetically Controlled Modules of Behavior. *Cell reports* **28,** 1814-1829.e6; 10.1016/j.celrep.2019.07.038 (2019).

117. Knutsen, P. M., Derdikman, D. & Ahissar, E. Tracking whisker and head movements in unrestrained behaving rodents. *Journal of neurophysiology* **93,** 2294–2301; 10.1152/jn.00718.2004 (2005).

118. Sofroniew, N. J., Cohen, J. D., Lee, A. K. & Svoboda, K. Natural whisker-guided behavior by head-fixed mice in tactile virtual reality. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **34,** 9537–9550; 10.1523/JNEUROSCI.0712-14.2014 (2014).

119. Kretschmer, F., Tariq, M., Chatila, W., Wu, B. & Badea, T. C. Comparison of optomotor and optokinetic reflexes in mice. *Journal of neurophysiology* **118,** 300–316; 10.1152/jn.00055.2017 (2017).

120. Mitchiner, J. C., Pinto, L. H. & Vanable, J. W. Visually evoked eye movements in the mouse (Mus musculus). *Vision Research* **16,** 1169-IN7; 10.1016/0042-6989(76)90258-3 (1976).

121. Newell, A., Yang, K. & Deng, J. Stacked Hourglass Networks for Human Pose Estimation. In *Computer Vision – ECCV 2016,* edited by B. Leibe, J. Matas, N. Sebe & M. Welling (Springer International Publishing, Cham, 2016), Vol. 9912, pp. 483–499.

122. Pérez-Escudero, A., Vicente-Page, J., Hinz, R. C., Arganda, S. & Polavieja, G. G. de. idTracker: tracking individuals in a group by automatic identification of unmarked animals. *Nature methods* **11,** 743–748; 10.1038/nmeth.2994 (2014).

123. Romero-Ferrero, F., Bergomi, M. G., Hinz, R. C., Heras, F. J. H. & Polavieja, G. G. de. idtracker.ai: tracking all individuals in small or large collectives of unmarked animals. *Nature methods* **16,** 179–182; 10.1038/s41592-018-0295-5 (2019).

124. Wu, X., Sahoo, D. & Hoi, S. C. H. Recent Advances in Deep Learning for Object Detection, 10/08/2019.

125. Schweihoff, J. F. *et al.* DeepLabStream: Closing the loop using deep learning-based markerless, real-time posture detection. *BioRxiv*; 10.1101/2019.12.20.884478 (2019).

126. Saunders, J. L. & Wehr, M. Autopilot: Automating behavioral experiments with lots of Raspberry Pis. *BioRxiv*; 10.1101/807693 (2019).

127. Travis, J. & Kring, J. *LabVIEW for everyone. Graphical programming made easy and fun.* 3rd ed. (Prentice Hall, Upper Saddle River, N.J., London, 2006, 2007).

128. Stockhausen, A. *et al.* Hard-wired lattice light-sheet microscopy for imaging of expanded samples. *Optics express* **28,** 15587–15600; 10.1364/OE.393728 (2020).

129. Wassie, A. T., Zhao, Y. & Boyden, E. S. Expansion microscopy: principles and uses in biological research. *Nat Methods* **16,** 33–41; 10.1038/s41592-018-0219-4 (2019).

130. Parekh, R. & Ascoli, G. A. Neuronal Morphology goes Digital: A Research Hub for Cellular and System Neuroscience. *Neuron* **77,** 1017–1038; 10.1016/j.neuron.2013.03.008 (2013).

131. Ferrante, M., Migliore, M. & Ascoli, G. A. Functional impact of dendritic branch-point morphology. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **33,** 2156–2165; 10.1523/JNEUROSCI.3495-12.2013 (2013).

132. Yi, G.-S., Wang, J., Deng, B. & Wei, X.-L. Morphology controls how hippocampal CA1 pyramidal neuron responds to uniform electric fields: a biophysical modeling study. *Scientific reports* **7,** 3210; 10.1038/s41598-017-03547-6 (2017).

133. Luo, W. *et al.* Supernova: A Versatile Vector System for Single-Cell Labeling and Gene Function Studies in vivo. *Scientific reports* **6,** 35747; 10.1038/srep35747 (2016).

134. Tyson, A. L. *et al.* A deep learning algorithm for 3D cell detection in whole mouse brain image datasets. *PLoS computational biology* **17,** e1009074; 10.1371/journal.pcbi.1009074 (2021).

135. Ibotta ML. *pure-predict. Machine learning prediction in pure Python* (Ibotta Inc., Denver, CO, USA, 2020).

136. Payne, H. L. & Raymond, J. L. Magnetic eye tracking in mice. *eLife* **6**; 10.7554/eLife.29222 (2017).

137. Coffey, K. R., Marx, R. G. & Neumaier, J. F. DeepSqueak: a deep learning-based system for detection and analysis of ultrasonic vocalizations. *Neuropsychopharmacology : official publication of the American College of Neuropsychopharmacology* **44,** 859–868; 10.1038/s41386-018-0303-6 (2019).

138. van Segbroeck, M., Knoll, A. T., Levitt, P. & Narayanan, S. MUPET-Mouse Ultrasonic Profile ExTraction: A Signal Processing Tool for Rapid and Unsupervised Analysis of Ultrasonic Vocalizations. *Neuron* **94,** 465-485.e5; 10.1016/j.neuron.2017.04.005 (2017).

139. Zala, S. M., Reitschmidt, D., Noll, A., Balazs, P. & Penn, D. J. Automatic mouse ultrasound detector (A-MUD): A new tool for processing rodent vocalizations. *PloS one* **12,** e0181200; 10.1371/journal.pone.0181200 (2017).

140. Kügler, S., Lingor, P., Schöll, U., Zolotukhin, S. & Bähr, M. Differential transgene expression in brain cells in vivo and in vitro from AAV-2 vectors with small transcriptional control units. *Virology* **311,** 89–95; 10.1016/S0042-6822(03)00162-4 (2003).

141. Shevtsova, Z., Malik, J. M. I., Michel, U., Bähr, M. & Kügler, S. Promoters and serotypes: targeting of adeno-associated virus vectors for gene transfer in the rat central nervous system in vitro and in vivo. *Experimental physiology* **90,** 53–59; 10.1113/expphysiol.2004.028159 (2005).

142. During, M. J., Young, D., Baer, K., Lawlor, P. & Klugmann, M. Development and optimization of adeno-associated virus vector transfer into the central nervous system. *Methods in molecular medicine* **76,** 221–236; 10.1385/1-59259-304-6:221 (2003).

143. Schindelin, J. *et al.* Fiji: an open-source platform for biological-image analysis. *Nature methods* **9,** 676–682; 10.1038/nmeth.2019 (2012).

144. Siu Kwan Lam *et al. numba/numba: Version 0.53.1* (Zenodo, 2021).

145. Fabian Pedregosa *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).

146. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585,** 357–362; 10.1038/s41586-020-2649-2 (2020).

147. Reback, J. *et al. pandas-dev/pandas: Pandas 1.2.5* (Zenodo, 2021).

148. McKinney, W. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference* (SciPy2010), pp. 56–61.

7 Methods

7.1 Mice

C57BL/6 mice were purchased from Charles River (Sulzfeld, Germany) and maintained on a 12-h light/12-h dark cycle with food and water always available. All experiments were carried out in accordance with the German animal protection law (TierSCHG), FELASA, and were approved by the animal welfare committee of the University of Bonn.

7.2 AAV production

AAV pseudotyped vectors (virions containing a 1:1 ratio of AAV1 and AAV2 capsid proteins with AAV2 ITRs) were generated as described [140,141]. Briefly, human embryonic kidney 293 (HEK293) cells were transfected with the AAV cis plasmid and the helper plasmids by standard calcium phosphate transfection. Forty-eight hours after transfection, the cells were harvested and the virus purified using heparin affinity columns (Sigma, St. Louis, MO) [142]. Purification and integrity of the viral capsid proteins (VP1-3) were monitored on a Coomassie-stained SDS/protein gel. The genomic titers were determined using the ABI 7700 real-time PCR cycler (Applied Biosystems) with primers designed for WPRE.

7.3 Surgical procedure

Viral injections were performed under aseptic conditions in two months old C57BL/6 mice.

For optogenetic closed-loop experiments, mice were initially anesthetized with an oxygen/isoflurane mixture (2 %–2.5 % in 95 % O_2). Afterwards, mice were fixed on a stereotactic frame and kept under a constant stream of isoflurane (1.5 %–2 % in 95 % O_2) to maintain anesthesia. Analgesia (0.05 mg/kg of buprenorphine; Buprenovet, Bayer, Germany) was administered intraperitoneal before the surgery, and Xylocaine (AstraZeneca, Germany) was used for local anesthesia. Stereotactic injections and implantations of light fiber ferrules were performed using a stereotactic frame (WPI Benchmark/Kopf) and a microprocessor-controlled minipump (World Precision Instruments, Sarasota, Florida). The viral solution (1:1:2; AAV-TRE-EGFP, Addgene #89875; AAV-M13-TEV-C-P2A-TdTomato, Addgene #92391; AAV-TM-CaM-NES-TEV-N-AsLOV2-TEVseq-tTA, Addgene plasmid # 92392) was injected unilaterally into the ADN. Viruses were produced as previously described. Animals were given Dexamethasone (0.2 mg/kg) to reduce swelling. For implantation, the skin on the top of the scalp was removed and the skull was cleared of soft tissue. Light fiber ferrules (Ø200 µm, 0.5 NA, Thorlabs) were implanted and fixed with a socket of dental cement. Loose skin around the socket was fixed to the socket using tissue glue (3M Vetbond). Directly after the surgery, animals were administered 1 ml 5 % Glucosteril solution. To prevent the wound pain, analgesia was administered on the three following days. Animals were left to rest for at least one week before starting handling. Experiments were conducted three weeks after surgery.

For Tetbow tracing experiments, mice were anesthetized with a mixture of Fentanyl (Rotexmedica, Germany), Midazolam (Rotexmedica, Germany), and Domitor (Orion Pharma, Finland) via intraperitoneal injection (i.p.; 0.05/5.0/0.5mg/kg). Analgesia was administered as mentioned above. Stereotactic injections of a viral solution (600 nl; 1:2:1:3; AAV-TRE-tdTomato-WPRE, Addgene #104112; AAV-TRE-EYFP-WPRE, Addgene #104111; AAV-TRE-mTurquoise2-WPRE, Addgene plasmid # 104110, AAV-Syn1-tTA, Addgene #104109; see also Supplementary Table D for a full injection scheme) was injected unilaterally into the CA3 Region of the hippocampus (r/c -2.1; l 2.5; d/v -2.25).

Viruses were produced as previously described. After the injection, the scalp was sutured (PERMA-HAND Silk Suture, Ethicon), and an antibacterial ointment (Refobacin, Almirall, Germany) was applied. Finally, a mixture of Naloxone (B.Brain, Germany), Flumazenil (B.Braun, Germany), and Antisedan (Orion Pharma, Finland) (1.2/0.5/2.5 mg/kg) was injected i.p. to end anesthesia. To prevent wound pain, analgesia was administered in the following three days. Mice were perfused, and brain samples were collected after 14 days.

7.4 Perfusion

Mice were anesthetized with a mixture of Xylazine (10 mg/kg; Bayer Vital, Germany) and ketamine (100 mg/kg; Bela-pharm GmbH & Co. KG, Germany). Using a peristaltic pump (Laborschlauchpumpe PLP33, Mercateo, Germany), the mice were transcardially perfused with 1× PBS followed by 4 % paraformaldehyde (PFA) in PBS. Brains were removed from the skull and post-fixed in 4 % PFA overnight (ON) at +4°C. After fixation, the brains were moved into PBS containing 0.01 % sodium azide and stored at +4°C until sectioning. Fixed brains were coronally sectioned using a vibratome (Leica VT1000 S) and stored at +4°C in PBS containing 0.01 % sodium azide.

7.5 Expansion of tissue samples

The expansion of tissue samples was adopted from protocols previously described [57,79,80]. Briefly, sections were incubated in 1 mM methyl-acrylic acid-NHS (Sigma Aldrich, Germany) linker. After washing (PBS), the sections were incubated in monomer solution (8.6 % sodium acrylate, 2.5 % acrylamide, 0.15 % N,N'-methylenebisacrylamide, and 11.7 % NaCl in PBS) for 1h, followed by 2h incubation at 37°C in gelling solution (monomer solution with addition of 4-hydroxy-TEMPO, TEMED and APS; resulting concentration: 0.01 %, 0.2 %, and 0.2 % respectively). After full gelation, the samples were then digested overnight at 37°C with Proteinase K in buffer solution (50 mM Tris, 1 mM EDTA, 0.5 % Triton-X100, 0.8M guanidine HCl, and 16U/ml of proteinase K; pH 8.0). After additional washing (PBS), the samples were either stored in PBS until imaging or expanded by additional washing with deionized water for three hours.

7.6 Imaging of brain sections

For optogenetic experiments, brain sections (70 µm) were labeled with DAPI (0.2 µg/ml). Overview images were acquired using a wide-field microscope (Zeiss AxioScan.Z1). Based on the overall expression and fiber placement, selected sections were imaged with a spinning disk microscope (VisiScope CSU-W1). Acquired z-stacks were used for quantification using FIJI [143]. Selection criteria for the quantification of Cal-Light labeling included the correct placement of the fiber ferrule above the target region as well as injection (Figure 9). Mice that did not match the criteria were only included in the evaluation and quantification of DLStream performance.

For Tetbow tracing experiments, the expanded samples were fixed on the bottom of a coverslip with poly-L-lysine to avoid displacement. To ensure stable expansion, the imaging chamber was filled with deionized water or PBS during imaging, depending on the imaging requirements. Imaging was performed with a custom light sheet fluorescence microscope as previously described [57]. However, because the samples were labeled with three different fluorescent proteins (tdTomato, EYFP, and mTurquoise2), the detection was further adapted. For this, the detection was conducted in two steps. First, EYFP and tdTomato were excited (488 nm and 561 nm, respectively), and emitted light was split onto two separate cameras. Second, mTurquoise2 was excited (405 nm) and detected by a camera. Each camera was preceded by an emission filter specific to the emitted light spectrum. This setup was necessary to refocus the light sheet due to a focus shift of the 405 nm laser and avoid displacement between color channels. Volumetric image acquisition was then realized by imaging in a mosaic fashion, where multiple image stacks were taken from each channel and stitched in postprocessing. Each image stack had a 10 % overlap with its neighboring image stacks to allow successful stitching. The axial stepsize was 0.5 µm, and the typical exposure time was 20 ms, while the field of view was 330µm with a pixel size of 0,163 µm.

7.7 Head direction-dependent optogenetic stimulation

Mice were put in a cylindrical white arena with a single cue (a black vertical bar, Figure 8 b). A black curtain enclosed the arena. A random point was chosen to reference head direction (0°, Figure 8 b red tape). The reference point was kept constant between experimental sessions and mice but was not visible to the mouse. To habituate the mice to the arena, each mouse was put into the arena for 30 min for two consecutive days, and reward pellets were placed randomly inside the arena at the 0, 10, and 20 min mark.

Experimental Group: During the experiment, light stimulation (488 nm, 15 mW; Laser OBIS LX/LS, controlled by OBIS LX/LS Single Laser Remote, Coherent Inc., Santa Clara, CA USA) was initiated whenever the mouse's head direction was within a 60° window around the reference point (± 30°). Light stimulation lasted at least 1 second or as long as the correct head direction was maintained, up to a maximum of 5 seconds. After each stimulus, further stimulation was discontinued for at least 15 seconds to avoid overheating brain tissue and in line with the originally published Cal-Light experiments [26]. Mice were allowed to investigate the arena over four consecutive days for 30 min sessions each day, during which the mice were stimulated with light depending on their head direction. Mice were perfused one day after the last session.

Yoked Group: In the yoked control group, mice were previously paired with another mouse from the experimental group. Each control animal received the exact same temporal stimulus as the paired experimental animal, decoupled from its own head direction. Mice were treated and ran the experiment in the same way as the experimental group in all other aspects.

No-Light Group: In the No-light control group, mice ran the experiment as all other groups but received no light stimulation.

7.8 Head direction analysis

Analysis was performed using custom python scripts. To determine whether light stimulation was precisely targeted to a particular window of angles, we calculated the mean resultant vector length for the distribution of stimulated angles, which measures the concentration of angles in a distribution. Lengths vary between 0 (the underlying distribution is uniform) to 1 (all angles in the underlying distribution are precisely identical). Thus, for stimulated angles, non-zero lengths close to 1 are expected. Notably, the distribution of stimulated angles may be biased by the mice's behavior – i.e., when the mouse, by chance, constantly faces the target head direction. To test against this possibility, null distributions were generated by randomly sampling angles from the entire distribution of angles explored by the animal. The number of samples was set to equal the number of stimulation angles. Angles were randomly sampled in this way 1000 times, and each time a mean resultant vector length was calculated. The null distribution comprised the 1000 means (note that null distributions were centered near 0). For each session, the resultant mean vector length was well above a 99 % cut-off of the null distribution, indicating that stimulation angle precision resulted from accurate posture detection rather than a bias in animal behavior.

7.9 Experimental setup

The corresponding arena was placed in a closable compartment with isolation from external light sources. A light source was placed next to the setup so that the arena was evenly lit. The camera was placed directly above the arena (Figure 6 a). During experiments, the compartment was closed to minimize any disrupting influences from outside. All devices were triggered using a NI 6341 data-acquisition board (National Instruments Germany GmbH, Munich) combined with the Python *nidaqxm* library. The board was connected via USB 3.0 to a PC (Intel Core i7-9700K @ 3.60GHz, 64 GB DDR4 RAM and NVidia GeForce RTX 2080 Ti(12GB) GPU). For the optogenetic experiment, an Intel Realsense Depth Camera D435 (Intel Corp., Santa Clara, CA, USA) was used at 848 x 480 and 30 Hz to enable reliable streaming at all times.

7.10 Hardware latency and detection accuracy during optogenetic stimulation

The latency between behavior detection and optogenetic stimulation was estimated by manually annotating videos of sessions from three different mice. For this, the recorded video was analyzed frame-by-frame. The frames between the event start (behavior-detection leading to stimulation onset) taken from the table-based output file and the visible onset of the Laser in the video were counted. All stimulation events during the above sessions were manually annotated to evaluate the false-positive detection rate during experiments (Figure 10). A detection was counted as false-positive when the annotator judged the mouse's posture (head direction) not inside the head direction window at the exact time of detection. Note that the accuracy of the pose estimation model is a major source of false detection; however, inaccurate event definitions can also lead to unintended stimulation events. Additional training of the network can increase the accuracy of the triggered stimulation.

7.11 Pose estimation using DLC

A 3-point tracking was used to estimate the mouse's position, direction, and angle using the nose, neck, and tail base as body parts of interest (Figure 3 b). Pose estimation models were trained using the DLC 1.11 framework [39]. First, 300 images of relevant behavior in the corresponding arena were annotated, and 95 % were used as a training set, with 5 % held back as a test set. Note that for some cases, a small number of test images (5 %, 15) might require further evaluation of the trained model to guarantee sufficient accuracy and generalization. Second, a ResNet-50-based neural network [75,103] with default parameters was trained for 500k iterations, and its performance was evaluated.

The same approach was used to benchmark DLStream's upper-performance limits, but images were labeled with either 9 or 13 body parts. The same training set was used to train several neural networks based on different architectures or depths (ResNet50, ResNet101 [75,103], MobileNetv2 [104]). Models were available through the DLC 2 framework with default parameters and trained for 500k iterations. After training, the networks were benchmarked within DLStream using a DLStream function (*python deeplabstream.py --dlc-enabled --benchmark-enabled*) with 3000 consecutive frames. Data were collected, and the average framerate and standard deviation were calculated for four different image resolutions (1280x1024, 640x512, 416x341, 320x256) available to the Basler acA1300-200um camera (Basler AG, Germany), which acquired frames at a rate of 172 Hz.

7.12 Behavior detection in DLStream

For behavior detection in the optogenetic experiment, the raw score maps were extracted from the deep neural network output, and the position of key points was calculated with custom scripts. First, body part estimation, similar to the approach utilized in DLC [39], was conducted by local maxima detection using custom image analysis scripts. The resulting pose estimation was then transferred into postures (*skeletons*). For this, each possible combination of body parts was investigated and filtered using a closest distance approach. DLStream detects estimated postures and compares them to relevant trigger modules for closed-loop control of experiments. Next, the pose estimation error was measured and compared to a human-labeled dataset (labeled by a single human annotator) to evaluate the pose estimation model. For this, a new image set was extracted from our optogenetic experiment sessions (n = 597). The average difference (Euclidean distance) between human annotation and pose estimation for each pose and resulting head direction angle were calculated.

Additionally, the false-positive/false-negative rate of hypothetical head direction triggers with differently sized angle windows (60, 50, 40, 30, 20, 10) was analyzed. To counter any non-uniform distribution of head direction angles, we averaged the rates for multiple ranges per bin (e.g., 0-60°, 60-120°, 120-180°) and calculated the standard deviation. See Figure 10 for details.

7.13 Machine learning-based classification in DLStream

The corresponding software toolkits were used to generate classifiers to evaluate machine learning classifiers based on B-SoiD [54] and SiMBA [76]. Example classifiers were integrated into DLStream and used as *trigger* modules in a simulated real-time video stream to evaluate their computation time, including feature extraction. A classifier pool of 3 parallel running classifier instances was used in combination with a simulated 30 Hz video stream using a prerecorded video. For real-time pose estimation during the measurement, DLC-based models were generated that matched the toolkit-specific requirements – e.g., number of tracked body parts. The pose estimation networks were trained in the same way as mentioned above.

To integrate and test SiMBA [76] classifiers in DLStream (see Supplementary Figure C a-b), an example pose estimation network, video, and a classifier were kindly provided by the original SiMBA authors [76]. In addition, the specific feature extraction script [66] was speed-optimized in collaboration with Simon Nilsson using the *numba* just-in-time compiler [144] that allows the translation of slow python algorithms into fast machine code. Finally, the classifier was then real-time optimized using *pure-predict* [135]. This open-source tool allows the translation of slow *scikit-learn*-based machine learning algorithms [145] to fast pure python code. Both variants of the classifier were used to estimate computation times.

To integrate and test B-SoiD [54] classifiers in DLStream, an example pose estimation network was trained according to the recommended body part configuration of B-SoiD. For this, animals were recorded in an open field arena from below, and example videos of their behavior extracted. Using the B-SoiD toolkit, the observed behavior was clustered, and a classifier was trained. Then, in collaboration with the original B-SoiD authors [54], the feature extraction script was integrated into DLStream [66]. Finally, the classifier and feature extraction script were used to estimate computation times.

Both feature extraction and classification computation time were measured for 1000 classification cycles to evaluate real-time capabilities. The resulting average time, including standard deviation, was then calculated (see Supplementary Table E).

7.14 Statistics and reproducibility

Paired t-tests were used for statistical comparisons of data. All data presented in the text are shown as the mean ± standard deviation. Uncorrected alpha (desired significance level) was set to 0.05 (* < 0.05, ** < 0.01, *** < 0.001). Sample sizes and numbers are indicated in detail in each figure caption and main text. Exclusion criteria, if applied, are specified in each corresponding method section.

8 Acknowledgments

When I started my doctorate, I was a pure-bred Biochemist with limited behavioral science and machine learning knowledge. However, in the course of this journey, I was able to grow beyond my hopes thanks to all the discussions and support that allowed me to develop and work with this cutting-edge technology.

I would like to thank my supervisor and mentor, Dr. Martin Schwarz, for giving me the opportunity and freedom to work on this thesis with all my heart. With his support, I tried out something completely different, and together we succeeded in it.

I would also like to thank my supervisors Prof. Dr. Ulrich Kubitscheck and Prof. Dr. Heinz Beck, for supervising my thesis and supporting me along the way.

It is essential to mention that this thesis was embedded in multiple collaborations with brilliant scientists:

Matvey Loshakov spent hours and hours with me coming up with solutions and ideas to design the initial version of DLStream. DLStream would not have been possible without him.

Dr. Laura Ewell was deeply involved in the design of the Cal-Light experiment and supported me in many steps along the way.

Laura Kück did the injections and implantations for Cal-Light and learned how to use DLStream as one of the first non-coding test subjects.

To all of you, I am very grateful that you were part of the DLStream Team. You helped me in many challenging situations, and I will pay it forward in kind.

The multicolor, hue-based segmentation approach would not have been possible without:

Dr. Jana Heysel, who developed the mentioned segmentation algorithm and imaged expanded tissue on her custom build LSFM.

Juan Edo Rodríguez Gatica, who did the tissue expansion and significant parts of the imaging.

It was a great experience working with you, and I look forward to future collaborations.

At last, I want to thank Prof. Dr. Sam Golden and Dr. Simon Nilsson from the Golden Lab, who provided example SimBA classifiers and helpful insights on the integration of SimBA based classifiers in DLStream.

9 Supplementary Information

9.1 Tables

Supplementary Table A – Available modules in the open-source version of DLStream

Type	Module	Use	Used in
Timer		Any time-dependent parameters	All experiments
Stimulation	NI – DAQ Board control	TTL controlled devices (e.g., Laser ON/OFF, Reward dispenser) Analog modulation of devices (e.g., Laser power)	Optogenetic experiment [1]
	GPIO control (Raspberry Pi)	TTL controlled devices (e.g., Laser ON/OFF, Reward dispenser)	Low budget setups
	GPIO control (Arduino)	TTL controlled devices (e.g., Laser ON/OFF, Reward dispenser)	Low budget setups
	Monitor/Screen display	Display visual stimulus (e.g., picture or video) on a screen	Conditioning experiment [1]
Trigger	ROI-based	If a body part or set of body parts is in or out of the region of interest (ROI)	Conditioning experiment [1]
	Direction-based (allocentric)	If a user-defined vector between body parts angle is within the defined window in relation to the reference point.	Published as an archetype
	Direction-based (Headdirection – allocentric)	If the head direction angle is within a defined window in relation to the reference point.	Optogenetic experiment [1]
	Direction-based (Screen)	Similar to allocentric direction trigger, but checks if the animal faces north, south, east, or west in the frame.	Conditioning experiment [1]
	Direction-based (Headdirection – egocentric)	If the egocentric head direction angle is within a defined window.	Published as an archetype
	Movement-based	If the animal (measured by a body part of choice) moves faster or slower than the threshold within a set time window.	Supplementary Material [1]
	Combinatio (Headdirection + ROI)	Example of a combination of multiple trigger modules. Checks whether the position and head direction of an animal are within the definition.	Published as an archetype
	ML-Classification	Set of machine learning-based behavior classification (SimBA [76], B-SOiD [54])	Published as an archetype
	Multiple Animals	Example of social behavior-based trigger module using multiple animal pose estimation.	Published as an archetype

Supplementary Table B – Available experiment modules in the open-source version of DLStream

Experiment Type	**Use**	**Used in**
Cal-Light	Head direction-dependent optogenetic stimulation of animals during Cal-Light paradigm.	Used in this thesis and published in Schweihoff et al. 2021
Conditional	A versatile experiment specifically created to allow conditional stimulation. It can be used with any trigger and can automatize behavior-dependent stimulation such as reward delivery/withdrawal for conditioning experiments.	Published as an archetype
Optogenetic	Experiment specifically designed for optogenetic paradigms. It holds additional parameters such as minimum/maximum stimulation time per event and maximum stimulation time in total.	Published as an archetype
Trial	Specifically designed to allow trial/task-based experiments. A primary *trigger* is used to initiate trials/task events in which an animal is presented with a *stimulation*. A secondary *trigger* checks if the animal succeeds in a pre-set time after/during the event (e.g., going to a reward location).	Published as an archetype
Classic Conditioning	Set of experiments for second-order conditioning paradigm including habituation to reward delivery, first conditioning, and transfer task.	Published in Schweihoff et al. 2021
Multiple Animal	Example experiment utilizing multiple animal pose estimation	Published as an archetype
Classification-based (supervised)	Example experiment incorporating machine learning-based, supervised behavior classification (SimBA [76])	Published as an archetype in collaboration with SimBA [76]
Classification-based (unsupervised)	Example experiment incorporating machine learning-based, unsupervised behavior classification (B-SOiD [54])	Published as an archetype in collaboration with B-SOiD [54]

Supplementary Table C – Example of DLStream output

The table is indexed by the frame ID used for pose estimation (neck, nose, tail base x+y). The experiment column holds information about the experiment. Status indicates whether the experiment was started (True), while Trial indicates whether a trial or stimulation was active during the frame (True, bold). The time column logs the inference time between frames. The CSV format uses a semicolon (";") as a delimiter to avoid confusion between German and American separation of decimals ("1,2" And "1.2") when importing the file.

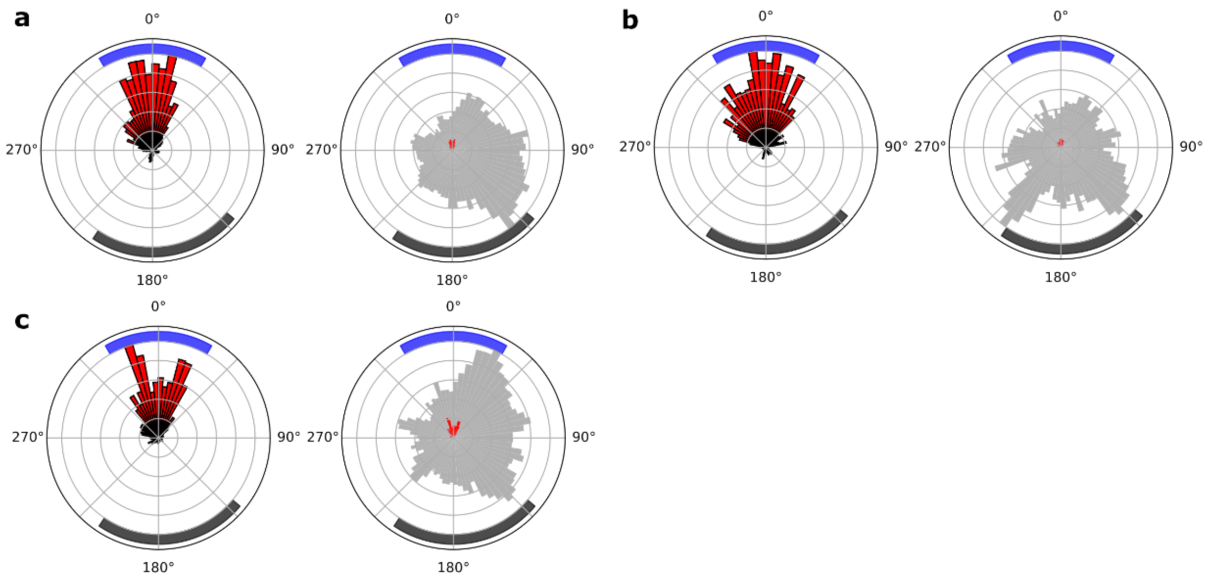
Frames	Animal 1						Experiment		Time
	Neck		Nose		Tail_base		Status	Trial	
	x	y	x	y	x	y			
1	48.45	43.89	45.45	43.89	51.45	29.89	False	False	0.0
2	45.45	43.89	51.45	45.79	55.35	32.91	True	False	0.033
3	44.13	46.91	49.45	41.11	57.65	35.79	True	**True**	0.066
4	45.25	42.11	45.55	42.77	55.45	29.49	True	**True**	0.099
5	49.26	44.89	48.25	43.99	50.33	29.89	True	False	0.132

Supplementary Table D –Tetbow injection scheme

Tetbow parameters optimized during development for hue-based segmentation of hippocampal neurons. The ratio of virus components and injection volume were evaluated on the resulting color diversity and label density by visual inspection based on confocal images of the predigested and digested samples. *Final mixture

Ratio				Injection Volume [nl]	Color Diversity	Label Density	Number of mice
tTA	mTurq	eYFP	tdTom				
1:500	1	1	1	300	+	---	1
1:100	1	1	1	300	+	--	2
1:50	1	1	1	300	+	-	2
1:10	1	1	1	300	++	+	2
	2	1	3	300	+++	+	3
				1000	+++	++	3
				600	+++	++	1
1:1	2	1	3	600	+++	++	2*

9.2 Figures



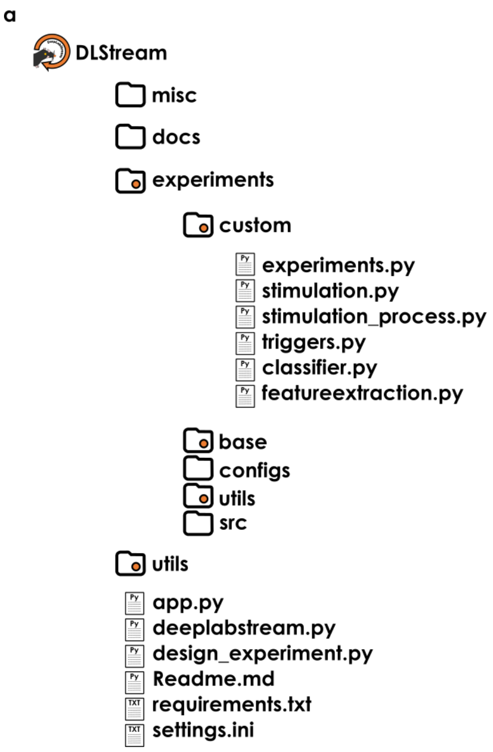
Supplementary Figure A - Examples of head direction angles during optogenetic light stimulation.

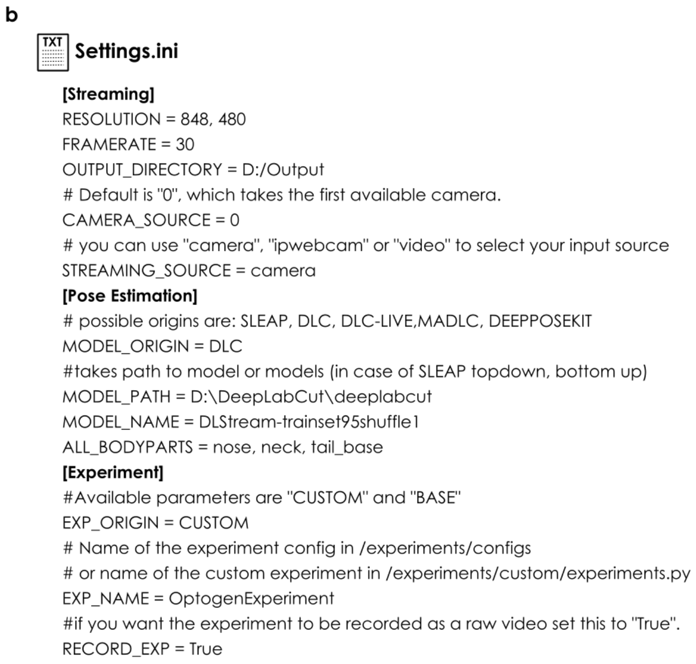
a-b, Left: Example radial histogram of all head directions (5° bins) during stimulation (red) within one session (normalized to the maximum value). Right: Radial histogram of all head directions during the whole session (grey) and during stimulation (red; normalized to the maximum value of the entire session). Rings represent quantiles in 20 % steps. Each panel shows a session from a different mouse.

c, Example radial histogram of all head directions (same representation as in **a-b**) from the same mouse shown in **a** in the next session. Note that the mouse is showing different distributions of head direction between sessions in both the stimulation events and the overall session, while the stimulation is mainly limited to the target window (thick blue arc)

9.3 DLStream Code and examples

9.3.1 DLStream package structure




b
Settings.ini

```
[Streaming]
RESOLUTION = 848, 480
FRAMERATE = 30
OUTPUT_DIRECTORY = D:/Output
# Default is "0", which takes the first available camera.
CAMERA_SOURCE = 0
# you can use "camera", "ipwebcam" or "video" to select your input source
STREAMING_SOURCE = camera
[Pose Estimation]
# possible origins are: SLEAP, DLC, DLC-LIVE,MADLC, DEEPPOSEKIT
MODEL_ORIGIN = DLC
#takes path to model or models (in case of SLEAP topdown, bottom up)
MODEL_PATH = D:\DeepLabCut\deeplabcut
MODEL_NAME = DLStream-trainset95shuffle1
ALL_BODYPARTS = nose, neck, tail_base
[Experiment]
#Available parameters are "CUSTOM" and "BASE"
EXP_ORIGIN = CUSTOM
# Name of the experiment config in /experiments/configs
# or name of the custom experiment in /experiments/custom/experiments.py
EXP_NAME = OptogenExperiment
#if you want the experiment to be recorded as a raw video set this to "True".
RECORD_EXP = True
```

Supplementary Figure B - Folderstructur of DLStream package

a, the Folder structure of the DLStream package available on GitHub. The package includes several scrips (text icon with “Py”; *.py) and text files (text icon with “TXT”; *.txt, *.md, or *.ini). Folders that include scripts are labeled with an orange circle. Scripts and files not relevant for the general structure and function of DLStream are not displayed.
b, Extract of the content of Settings.ini. The file contains all information that is used to start individual experiments when running the script app.py. The section [Streaming] configures camera-specific parameters such as resolution and framerate. The section [Pose Estimation] is used to select and load a pose estimation model from the available architectures (DLC [39,45,46], SLEAP [40,43], DeepPoseKit [41]). The section [Experiment] is used to select an *experiment* module from *experiments/custom/experiments.py* or *experiments/base/experiments.py* that contain custom or predefined experiments.

The DLStream package is structured so that modules are separated into scripts (e.g., *experiments* in *experiments.py*; Supplementary Figure B a). This structure has the advantage that customized modules can be easily implemented and imported between scripts, while the main functions of *deeplabstream.py* remain untouched. Using the file *settings.ini* (Supplementary Figure B b), users can select the name of an *experiment* module, and the *experiment* will be automatically loaded when *app.py* is launched. The script a*pp.py* opens the GUI of DLStream, including the main process of *deeplabstream.py* (see chapter 4.1.3), and gives users a convenient way of interacting with DLStream during experiments. *Settings.ini* also contains configuration parameters for the camera and pose estimation settings. To load a specific pose estimation model, users specify the model's origin, the path to the model, and the model name. The model can then be launched using the GUI (see chapter 4.1.3). Independent of the model origin, the resulting pose estimation is transformed into a *skeleton* that can be interpreted by any *experiment* and *trigger* module in DLStream.

9.3.2 Experiment module for the optogenetic experiment

The following is an extract of the code used for the head direction-dependent labeling of active neurons. It includes several simplifications and explanations. The original code is fully published at https://github.com/SchwarzNeuroconLab/DeepLabStream [66].

```
DeepLabStream
© J.Schweihoff, M. Loshakov
University Bonn Medical Faculty, Germany
https://github.com/SchwarzNeuroconLab/DeepLabStream
Licensed under GNU General Public License v3.0

class OptogenExperiment:
   def __init__(self):
      ...
      self._point = POINT
      self._start_angle, self._end_angle = 30
      self._intertrial_timer = Timer(15)
      self._experiment_timer = Timer(1800)
      ...
      self._max_trial_time = 5
      self._min_trial_time = 1
      self._max_total_time = 600
       ...
```

1 - Initializing

The optogenetic experiment is initialized as a python class and has several initial parameters (Code 1 - Initializing). At its core, DLStream calls the *experiment* with every new pose estimation – i.e., every frame – and passes the pose estimation to the *experiment.* The *experiment* then passes the pose estimation to a *trigger* module and, if the behavior was detected, activates the *stimulation* module (see Figure 6 d for reference).

The parameters $self.\_point$, $self.\_start\_angle$, and $self.\_end\_angle$ define the target window of 60° around the reference point (*POINT*). Two *timer* modules are initialized with 15 sec and 1800 sec duration. The 15-sec timer $self.\_intertrial\_timer$ acts as an inter-stimulus timer and inhibits any behavior-dependent stimulation during the inter-stimulus time. The $self.\_experiment\_timer$ is measuring the duration of the entire session and stops the

```
def check_skeleton(self, frame, skeleton):
        if self._experiment_timer.check_timer():
            if self._total_time >= self._max_total_time:
                # check if total time to stimulate per experiment is reached
                print("Ending experiment, total event time ran out")
                self.stop_experiment()
            else:
                # if not continue
                if not self._intertrial_timer.check_timer():
                    # check if there is an intertrial time running right now, if not continue
                    # check if the headdirection angle is within limits
                    _ , angle_point = angle_between_vectors(
                        *skeleton["neck"], *skeleton["nose"], *self._point)
```

2 - Check skeleton

experiment after 1800 sec (30 min). The parameters $self.\_\max\_trial\_time$ and $self.\_\min\_trial\_time$ control the maximum and minimum duration of light stimulation during the experiment while the $self.\_\max\_total\_time$ acts as a maximum total stimulation threshold. Each time the *experiment* is passed a set of pose estimated body parts (*skeleton*) using the function *check_skeleton()*, the *skeleton* is passed to the *trigger* module. To simplify this, the code example includes the relevant calculations that are integrated into the head direction *trigger* module as plain code (Code 2- Check skeleton).

First, the *experiment* checks whether the total duration of the experiment has run out using the *timer* module $self.\_experiment\_timer$. The experiment is stopped when the timer ran out. Next, the inter-stimulus timer is checked. If the inter-stimulus time was activated and has not yet run out, the *experiment* is skipping any further operations – i.e., a behavior-dependent stimulation is inhibited. Otherwise, the head direction angle is calculated – usually within the *trigger* module, and a preset condition is checked. In this *experiment*, the output of the *trigger* module is TRUE – i.e., the behavior was detected – when the head direction angle is equal or between -30 and +30° (Code 3 - Behavior detection).

If a *stimulation* event has not been started yet, the *experiment* will call the *stimulation* module. In this *experiment,* it is a simple ON signal to a laser. The function *laser_switch()* is a high-level interface for the NI DAQ-Board Digital output port and sends a TTL signal to a remote control for the laser. For more advanced *stimulation* modules, *experiments* are equipped with an additional process handling only *stimulation*. This parallel processing step is necessary to process both new pose estimation with each frame and continuously run the experiment.

```
            if self._start_angle <= angle_point <= self._end_angle:
                if not self._event:
                    # if a stimulation event wasn't started already, start one
                    print("Starting Stimulation")
                    self._event = True
                    # and activate the laser, start the timer and reset the intertrial timer
                    laser_switch(True)
                    self._event_start = time.time()
                    self._intertrial_timer.reset()
```

3 - Behavior detection

However, in this case, the remote control of the laser and the DAQ boards are handling the continuous stimulation downstream.

With each new event, the inter-stimulus timer $self.\_intertrial\_timer$ is reset, and the start of the event is timed to calculated the minimum and maximum stimulation time.

If the head direction angle is within the target window, but a stimulation event is already running, the *experiment* checks whether the maximum stimulation duration per event was reached (Code 4 - Light stimulation **I**). If the maximum duration was reached, the laser is turned OFF, the duration of the stimulation is recorded, and the inter-stimulus timer is started. If the maximum duration was not reached, the stimulation continues. Contrary, if the head direction angle is not within the target window, but a stimulation event is already running, the *experiment* checks whether the minimum stimulation duration per event was reached (Code 5 - Light Stimulation I). If the minimum duration was reached, the laser is turned OFF, the

```
        else:
            if time.time() - self._event_start <= self._max_trial_time:
                # if the total event time has not reached the maximum time per event
                pass
            else:
                # if the maximum event time was reached, reset the event,
                # turn off the laser and start intertrial time
                print("Ending Stimulation, Stimulation time ran out")
                self._event = False
                laser_switch(False)
                trial_time = time.time() - self._event_start
                self._total_time += trial_time
                print("Stimulation duration", trial_time)
                self._intertrial_timer.start()
```

4 - Light stimulation I

duration of the stimulation is recorded, and the inter-stimulus timer is started. If the minimum duration was not reached, the stimulation continues. The *experiment* continues until the $self.\_experiment\_timer$ has run out.

```
    else:
        # if the headdirection is not within the parameters
        if self._event:
            # but the stimulation is still going
            if time.time() - self._event_start < self._min_trial_time:
                # check if the minimum event time was not reached, then pass
                pass
            else:
                # if minumum event time has been reached, reset the event,
                # turn of the laser and start intertrial time
                print("Ending Stimulation, angle not in range")
                self._event = False
                laser_switch(False)
                trial_time = time.time() - self._event_start
                self._total_time += trial_time
                print("Stimulation duration", trial_time)
                self._intertrial_timer.start()
```

5 - Light stimulation II

9.3.3 Example trigger module

A *trigger* module is an object that is specifically created to check whether a specific predefined condition is true. Its input is the *skeleton* (pose estimation of all body parts in the current frame), and its output is a binary classification (TRUE/FALSE) whether a preset

```
class RegionTrigger:
    def __init__(self, region_type: str, center: tuple, radius: float, bodyparts,
        debug:  bool = False):
        self._roi_type = region_type.lower()
        region_types = {'circle': EllipseROI, 'square': RectangleROI}
        self._region_of_interest = region_types[self._roi_type](center, radius, radius)
        self._bodyparts = bodyparts
```

6 - Region Trigger module I

condition was met. The *trigger* module can incorporate any calculation, condition, or algorithm to classify a single pose estimation or a sequence of pose estimations. A set of example *trigger* modules that were published with DLStream is shown in Supplementary Table A.

The basic architecture of a *trigger* module is shown in Code 6-7. A region of interest (ROI) *trigger* module comes with a simple set of parameters. The type of region (rectangle, circle, or ellipse), a center coordinate, and a radius or length/width parameter for the ROI. Any body part specified during the initialization will be tested for the condition when the function *check_skeleton()* is called by the *experiment* (see Supplementary Information 9.3.2) – i.e., if the body part is within the defined ROI. Depending on the result (TRUE or FALSE), a *response_body* is created to visualize the *result* in the live video stream. For example, with a circular ROI, the response body consists of a circle with the radius and center of the ROI. Its color is based on *result*. Red for FALSE and green for TRUE (see Figure 7).

```
    def check_skeleton(self, skeleton: dict):
        # check whether bodypart is in ROI
        bp_x, bp_y = skeleton[self._bodyparts]
        result = self._region_of_interest.check_point(bp_x, bp_y)
        # The following creates the response_body that is visualized on the screen
        color = (0, 255, 0) if result else (0, 0, 255)
        if self._roi_type == 'circle':
            response_body = {'plot': {'circle':
                    dict(center=self._region_of_interest.get_center(),
                    radius=int(self._region_of_interest.get_x_radius()),
                    color=color)}}
        response = (result, response_body)
        return response
```

7 - Region Trigger module II

The *response_body* can also take other shapes or plot information as text, depending on the individual design of the *trigger*.

9.3.4 Example stimulation module

```
def example_protocol_run(condition_q: mp.Queue):
  current_trial = None
  dmod_device = DigitalModDevice('Dev1/PFI0')
  while True:
    if condition_q.full():
      current_trial = condition_q.get()
    if current_trial is not None:
      show_visual_stim_img(img_type=current_trial, name='inside')
      dmod_device.turn_on()
    else:
      show_visual_stim_img(name='inside')
      dmod_device.turn_off()

    if cv2.waitKey(1) & 0xFF == ord('q'):
      break
```

8 - Stimulation module I

Stimulation modules, while straightforward to understand, require additional levels of code to work as required. Generally, they heavily depend on individual setups and experiment design. The following is an explanation of the fundamental basics of any *stimulation* module in DLStream.

In principle, a stimulation is triggered and activates a predefined cascade of events (see Supplementary Information 9.3.2). The core of a *stimulation* module runs parallel to the *experiment*, so it does not stop or slow down the main process (pose estimation and behavior classification). As stimulation cascades might be engaged for a longer time, multi-processing is necessary. If all computations were included in a single process, any stimulation event would block all further progress until it is completed.

The underlying architecture is split into separate scripts (refer to Supplementary Information 9.3.1). The script *stimulation.py* contains the actual stimulation. *show_visual_stim_img*(), for example, creates a window and displays a preset image on a screen. This function can switch between background and stimulation images on a screen visible to the animal from inside the arena. The functions *turn_on()* and *turn_off()* control a device connected via a control board (see Supplementary Table A) that sends a digital trigger (TTL) signal. A version of this function is used in the optogenetic experiment to toggle a laser (see Supplementary Information 9.3.2).

The script *stimulation_process.py* is the multi-process protocol that orchestrates the stimulation (Code 8 - Stimulation module **I**). In principle, a connection (queue) is built between the main DLStream process and the experimental protocol, controlling the stimulation event. This connection is straightforward and can only contain a single argument at a time. Once the *trigger* module detects a behavioral expression, the *experiment* passes an activation signal through the connection. The stimulation protocol, once started, remains in an endless loop. With every iteration of the loop, the protocol checks whether any new input came through the connection. If so, the stimulation event is initialized, and a preset cascade will be run. In the above example (Code 8 - Stimulation module **I**), the stimulation event displays a visual stimulation (image on a screen visible to the animal). Afterward, it activates a connected device (e.g., a reward dispenser). If the stimulation event is over, the protocol will display a background image and turn the device off. A similar protocol was used in Schweihoff et al. 2021 [1].

9.3.5 Adapting an existing experiment

The following is a short version of the complete instructions and tutorials available at https://github.com/SchwarzNeuroconLab/DeepLabStream/wiki.

As previously stated, DLStream *experiments* are designed with sequences of modules (*timer, stimulation, trigger*) that enable the autonomous conduction of behavior-dependent experiments. Thus, depending on the paradigm, it might be necessary to test several behavioral expressions within the same basic experiment. The optogenetic experiment, for example, could be used in combination with any behavioral expression to label active neurons with Cal-Light.

To change a *trigger* module, change the head direction trigger, $self.\_trigger$, to the *trigger* module of choice (Code 9 - Changing the trigger module; Supplementary Table A). When changing any module, it is essential to verify that all necessary parameters are included in the initialization. In this case, the region of interest trigger, *RegionTrigger,* needs a type of region (rectangle, circle, or ellipse), a center, and a radius or length/width parameter (see also Supplementary Information 9.3.3).

```
class OptogenExperiment:
    def __init__(self):
        …
        self._point = POINT
        ~~self._angle = 30~~
        ~~self._trigger = HeaddirectionTrigger(self._angle, self._point)~~
        self._trigger = RegionTrigger  (region_type = „circle“,center = self._point,  radius = 30,
                                                 bodyparts = [„nose“])
        self._intertrial_timer = Timer(15)
        self._experiment_timer = Timer(1800)
        self._max_trial_time = 5
        self._min_trial_time = 1
        self._max_total_time = 600
        …
```

9 - Changing the trigger module

Additionally, it is necessary to specify which body parts should be included in the behavior detection. For example, the *RegionTrigger* module initialized in this experiment (Code 9 - Changing the trigger module) would detect whenever the animal’s nose point is within a 30 px radius from the center.

As stated in chapter 9.3.2, a typical *experiment* passes the pose estimation (*skeleton*) from every frame to the *trigger* module. The generic way of doing this is shown in Code 10 - Engaging the trigger **module**. The *trigger* module outputs a binary classification (True/False) if the behavioral expression of interest was present in the current frame. This Input/Output

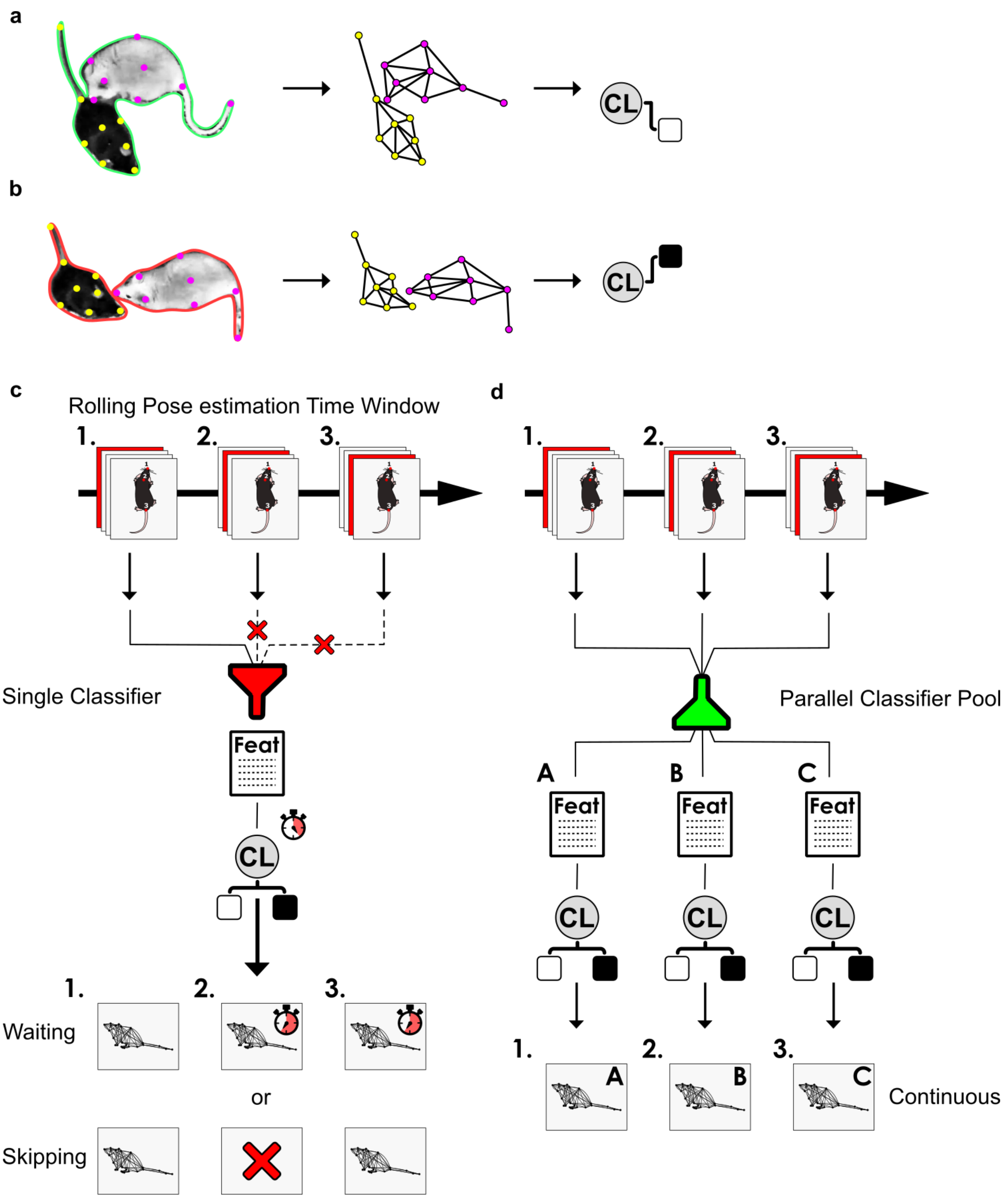
```
def check_skeleton(self, frame, skeleton):

    if self._experiment_timer.check_timer():
        if self._total_time >= self._max_total_time:
            # check if total time to stimulate per experiment is reached
        else:
            # if not continue
            if not self._intertrial_timer.check_timer():
                # check if there is an intertrial time running right now, if not continue
                # check if the headdirection angle is within limits
                result, response = self._trigger.check_skeleton(skeleton=skeleton)
                if result:
                    # if the trigger returns true
                else:
                    # if the trigger returns false
```

10 - Engaging the trigger module

behavior is fundamental to all *trigger* modules. It allows the exchange of *triggers* in the initial step (Code 9 - Changing the trigger module) of an *experiment* – i.e., independent of the type of *trigger* module, the input is always *check_skeleton(skeleton),* and the output is always TRUE or FALSE. Additionally, the *trigger* module outputs a *response* body that can be used to visualize the output on the live stream.

9.3.6 Feature extraction and classification in DLStream



Supplementary Figure C - Real-time classification in DLStream

a, Cutout of example frame showing anogenital approach behavior classification with SimBA. The pose estimation of the two mice (colored dots, left) is used to extract features (middle). The features are fed into the classifier (right), and a binary classification is computed (white square, green border around mice) and detects the behavioral expression of interest.
b, Cutout of example frame showing different behavior classified with them the same classifier as in **a**. The binary classification is computed (black square, red border around mice) and does not detect the behavioral expression of interest.

c, Schematic representation of a *trigger* module using a single classifier. The pose estimation sequence is updated with every new frame in a rolling window approach (Rolling Pose estimation Time Window, top). Time windows (stack of mice with red dots) are fed into the *classifier* module where features are extracted (document symbol, "Feat") and used as input for a ML-classifier (CL). If the classification process takes longer than the pose estimation of the next frame (stopwatch with red zone), a computational bottleneck is formed (red funnel). Any behavior classification (net representation of mouse, bottom) will have an additional latency ("Waiting") – i.e., the overflow until the classification is done. Alternatively, time windows need to be skipped to keep up with the real-time requirement ("Skipping").
d, schematic representation of a *trigger* module using a parallel classifier pool. The pose estimation sequence is updated with every new frame in a rolling window approach (Rolling Pose estimation Time Window, top). Time windows (stack of mice with red dots) are fed into the next idle *classifier* module instance where features are extracted (document symbol, "Feat") and used as input for the ML-classifier (CL). If the classification process takes longer than the pose estimation of the next frame (stopwatch with red zone), the next idle classifier instance is engaged (green funnel). Suppose the pool size exceeds the classification time divided by the pose estimation time by at least one. In that case, any additional unexpected computational load can be compensated so that an idle instance can readily classify every new time window. The resulting behavior classification (net representation of mouse, bottom) is continuous, and unused classifier instances remain idle until necessary.

DLStream utilizes a multi-process pool of classifiers to work in parallel. A *trigger* module built with a machine learning classifier initializes a pool of classifier instances and feeds in a pose estimation sequence (time window; Supplementary Figure C). The time window is continuously updated with each new pose estimation so that classification is based on a rolling window approach rather than a discrete binning.

To allow real-time classification with stable output times, multi-processing pools are required to compensate for occasional increased computational loads and general slow computation. For example, suppose the classification of a single classifier instance has a higher processing time than the pose estimation of the next frame. In a single classifier case, the classification of the next pose estimation window would be delayed until the classifier is ready. This would either increase the latency between pose estimation and behavior-dependent stimulation or reduce the detection rate because old, unclassified time windows are skipped in favor of the most recent window.

A solution is the integration of a multi-process pool that works parallel but asynchronous. In that case, whenever a new time window is ready, and the previous classifier instance is busy, a new classifier instance from the pool is engaged, and classification continues without overhang. This way, a *trigger* module based on ML classification has a maximum latency of one classification cycle (including feature extraction), and no additional latency or skipped frames are encountered (Supplementary Figure C).

Additionally, the classifier and feature extraction can be further optimized for real-time applications. For this, the feature extraction script, based originally on easy-to-use Python packages like *NumPy* [146], *pandas* [147,148], and *scikit-learn* [145], can be translated into fast compilable machine code using *numba* [144], considerably decreasing computation time (see Supplementary Table E). An optimized version of the feature extraction code used in SiMBA was developed in collaboration with Simon Nilsson and integrated into DLStream [66]. However, the same principle can be applied to any feature extraction script. Finally, classifiers based on *scikit-learn* [145], such as SiMBA or B-SoiD, can be real-time optimized using *pure-predict* [135] to translate the classifiers into pure Python-based versions that allow increased performance (see Supplementary Table E). With all three optimization steps combined, the effective integration of complex behavior classification into DLStream is possible with minimal additional latency.

Supplementary Table E - Classification and feature extraction performance

The Computation time of different classifiers and feature extraction scripts (FeatEx) as described in Methods 7.13.

Classifier Origin	FeatEx Origin	Extraction [ms]	Classification [ms]
B-SoiD	B-SoiD	38.25 ± 3.20	22.88 ± 4.36
SiMBA	SiMBA	235.36 ± 4.87	113.39 ± 5.72
pure-predict SiMBA	SiMBA	235.56 ± 4.72	33.71 ± 4.79
Simba	*Numba* optimized FeatEx	0.09 ± 0.69	114.04 ± 5.98
pure-predict SiMBA	*Numba* optimized FeatEx	**0.09 ± 0.69**	**9.44 ± 2.19**