Association Between Stereotyped Facial Motor Activity And Categorized Neuronal Population Dynamics At The Anterior Primary Motor Cortex Of Head Fixed Mice.

Masterarbeit
für die Prüfung zum Master of Science (M. Sc.) Biochemie
eingereicht beim Prüfungsausschuss für den
Masterstudiengang Biochemie
der Fakultäten für Biowissenschaften und Chemie und für Geowissenschaften
Ruprecht-Karls-Universität Heidelberg

Tristan Wiessalla
geboren in Aachen

03. August 2020

Declaration

This thesis was written from 27th January 2020 to 3rd August 2020 at Dr Robert Prevedel's research group at the European Molecular Biology Laboratory (EMBL), Heidelberg.

I declare that I wrote this thesis on my own and without any illicit external help. All citations from publications in this thesis are disclosed and clearly identified as such.

First examiner: Robert Prevedel
Second examiner: Jochen Wittbrodt
Defense committee member: Juan Carlos Boffi

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe verfasst habe und dass alle wörtlich oder sinngemäß aus Veröffentlichungen entnommenen Stellen dieser Arbeit unter Quellenangabe einzeln kentlich gemacht sind.

Erster Prüfer: Robert Prevedel
Zweiter Prüfer: Jochen Wittbrodt
Prüfungsbeisitzer: Juan Carlos Boffi

Heidelberg, den 30.07.2020

Acknowledgements

I would like to thank my mentor Juan Carlos Boffi for the guidance he provided me with when I got started in the field of Theoretical Neuroscience. I am grateful to Robert Prevedel and the rest of the Prevedel group at EMBL Heidelberg for giving me the opportunity to write this thesis and providing helpful discussion and vital input. I learned a lot writing this thesis. Thanks to Tom Boissonet and Moritz Wachsmuth-Melm, who also wrote the LaTeX class formatting this document, for getting me started with more intricate details of Coding. Finally I thank Juan, Benedikt Wimmer and Judith Notbohm for proofreading and suggestions.

Abstract

Correlation of neural activity with observed motor behavior is the central challenge of systems neuroscience. Traditionally, motor behavior in response to a presented stimulus and simultaneous neural activity have been recorded to establish that correlation. Recently it was shown that spontaneous, unstimulated movements are the cause of major parts of the recorded neural activity, partly obscuring the influence of stimulated movements[84, 101]. The mouse face exhibits many movement patterns, such as sniffing and whisking, that are linked to central pattern generators in the brainstem and thus mostly unaffected by stimuli. This makes it possible to investigate unstimulated movements with simple camera recordings of the face.

Here, I established an automated program library, FaceCat, to extract stereotyped movement patterns from spontaneous face movement of head-fixed mice. FaceCat was designed as the central tool in the Prevedel group's project to characterize the population dynamics in primary motor cortex in response to spontaneous face movements. This will allow further investigation of the cortico-bulbar pathway that controls muscles upwards of the neck. One recent study shows plasticity in population activity of primary motor cortex when mice where trained on a temporally precise stimulated licking task[66]. FaceCat makes it possible to detect and study similar characteristics in the control of face movement.

Preliminary applications of the algorithm in Juan Carlos Boffi's experiments show that FaceCat is able to detect and classify a decrease in the number of precise coordinated face movements in mice with an ablated primary motor cortex. This indicates that FaceCat is a valid approach to quantitatively study dexterous motor control of on-going, untrained, spontaneous behaviors in the mouse face.

Contents

Part I.

Introduction

1. Motivation: The Challenge of Systems Neuroscience

1.1. Correlation of Behavior With Neural Activity

Understanding brain function on a comprehensive scale is a long-established aim in Neuroscience. Systems Neuroscience, the branch of the field dedicated particularly towards this aim investigates neural circuits from the molecular to the cognitive level, their interconnectivity and the processes that govern sensory integration and decision-making. Systems Neuroscience commonly uses the quantification of motor behavior as a readout for brain activity. The main challenge is how to correlate acquired behavioral information with the activity of as many neurons as possible. Adapting this definition, early galvanic stimulation experiments can be thought of the first attempts to establish that link.

1.1.1. Brief History: From Somatotopy to Integrated Circuits

Historically, these stimulation experiments often focussed on stimulation of the motor cortex (M1) of anesthetized animals, since its behavioral output could be monitored with the bare eye [45, 47, 89] Starting with these early experiments, the classical view of the somatotopical brain mapping of M1 was established. Analogous to somatosensory perception, this view claims that subregions of M1 exert direct control over single or a small group of muscles, essentially steering the motor behavior of a certain part of the body FIGURE 1.1.

With an increasing number of neuroanatomical studies on the projection tracts of M1 and extensive input tracing of the spinal cord, this view was increasingly challenged [114]. Not only were the direct M1 to muscle connections found to be very rare, but also the general assumption of one area of the brain performing only one specific task had to be reassessed. Today new methods such as multi-photon deep tissue imaging of the brain, multichannel silicon probes, functional magnetic resonance imaging (fMRI), magneto-/ electro-encephalograms (MEG/EEG), electrocorticography ECoG

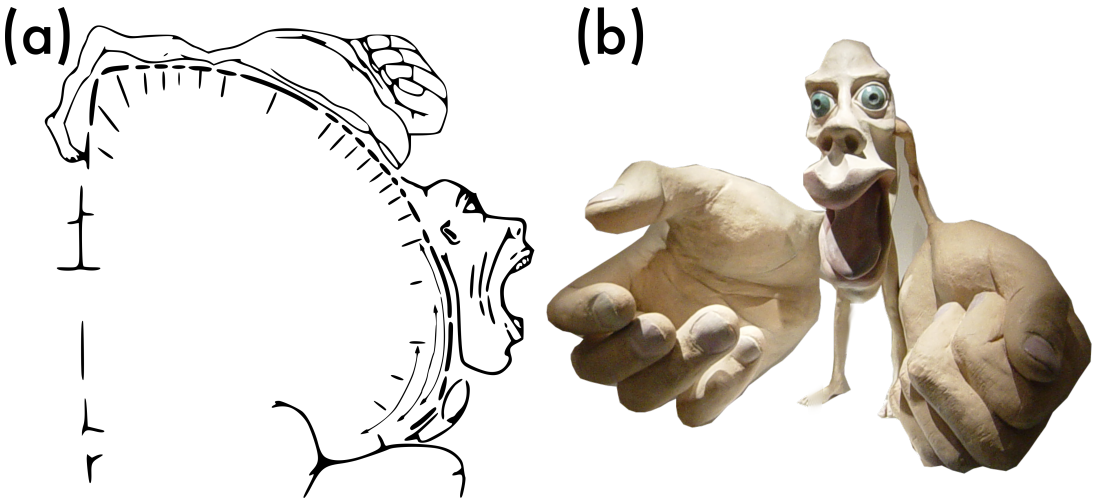

FIGURE 1.1.: Depictions of somatomotory homunculi. (a) A coronal section of the human brain. Body parts are located at the cortical areas that control them. (b) A somatomotory homunculus plastic. Here, body parts are scaled based on the size of their cortical representation

or functional ultrasound imaging aim at understanding neural populations and their interactions, allowing to view the brain as an integrated whole.

1.1.2. Analytical Ways to Establish Relationship Between Behavior and Neural Activity

Those methods pose new challenges in the overall amount of data they generate and the high dimensionality that comes with it. This only intensifies the challenge of Systems Neuroscience: How can the neural data be correlated with the observed behavior? Thus, analysis of the neuronal output analysis evolved analogously to experimental recording techniques.

HAXBY et al [57] define 4 major decoding/encoding strategies that sum up the main trends in neural output analysis:

1. Multivariate pattern (MVP) classification uses machine learning methods to define decision boundaries in a neural representational space that best distinguish a set of response vectors for one brain state from others. A neural representational space is a mapping of the neural response pattern features (for example single neuron activity) to dimensions of representational vectors.

2. Representational similarity analysis (RSA) quantifies the similarity between response vectors as distances in the representational space.

3. Stimulus-model-based encoding predicts the location of the neural response vector for a new stimulus on the basis of the coordinates of that stimulus in a stimulus feature space.

4. Stimulus-model-based decoding tests whether the encoding-based prediction allows correct classification of neural response vectors to new stimuli.

In the case of experiments targeting M1, approach 4 is the most promising one; for simple motor experiments it allows to answer the question: Can the neural response that was recorded be mapped to the representation of the motor behavior that was observed? To answer this question, one first has to decide on methods to extract representations of both, stimulus and behavior. After that, the relationship between those two representations needs to be assessed. Traditionally, various forms of forward and reverse correlation have been used to do this. JONES et al set the precedence to use reverse correlation metrics to obtain information about the stimulus preceding a given spike, nowadays called the spike triggered average (STA) [64] FIGURE 1.2.

Although still widely used, they suffer from some limitations, such as the assumption of linearity, so that more recently various approaches have been established that aim at overcoming some of the limitations of simple correlation. The main disadvantage is the need for an inter-trial repeatability of the stimulus. We aimed at recording unstimulated, natural face movements of head-fixed mice, preventing us from sorting and averaging motor responses based on stimulus. Alternative analysis methods include tensor factorization [26, 63] and information theory metrics [12], especially mutual information [27]. Regarding the general challenge of detecting the causality behind the correlation of measured behavior and neural activity, unstimulated motor activity has recently become the focus of attention, after it was shown that spontaneous face movements are drive neural activity in many parts of the brain [101] and that spontaneous movements may account for considerable trial-by-trial variance and also mimic or overshadow the cognitive processes that are commonly studied in trial-averaged data [83].

1.1.3. Outline

The motor behavior we assessed is unstimulated coordinated movement of the anterior parts of the mouse face (that is nose, snout, whisker pad). In the following introduction three major points will be presented in some detail: a) why it can be difficult to study this kind of behavior, especially in animals, b) the underlying neuroanatomy and c) computational methods/foundations to extract those behaviors

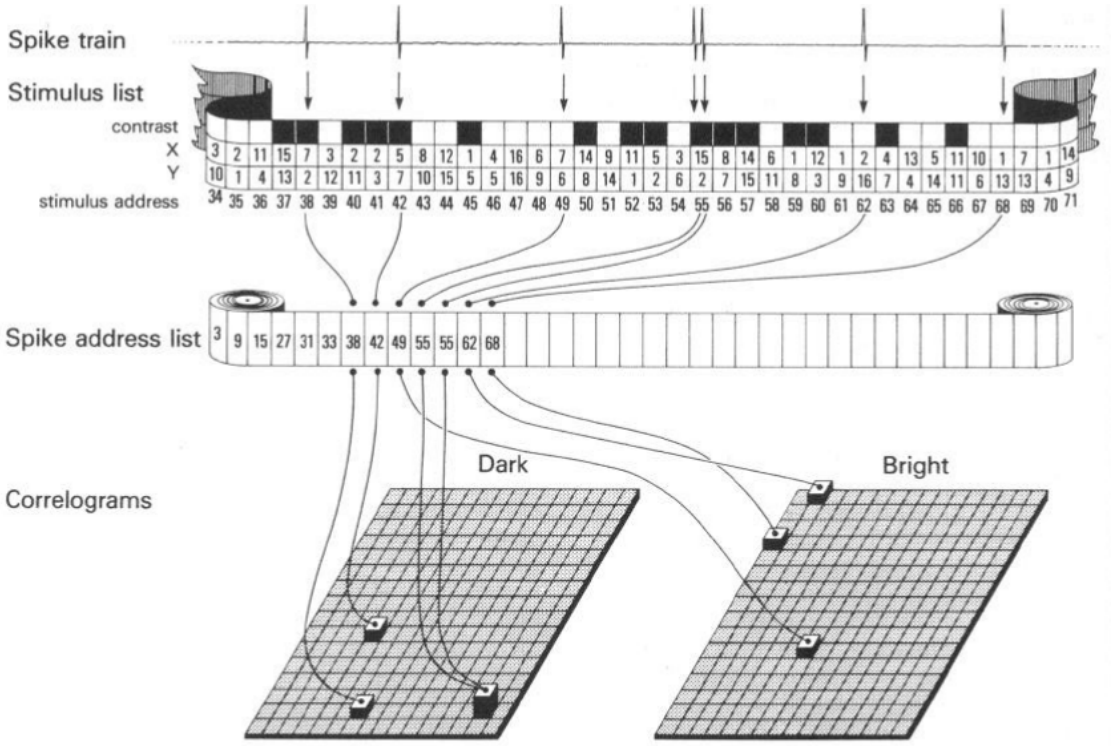

FIGURE 1.2.: Reverse correlation. Recorded spikes and presented stimuli (location in 2D and valence) are stored in two separate lists. The combined information about the stimulus is used to assign an address to each spike. From that, reverse correlograms are generated that decode location and valence of the stimulus that was presented when a spike was recorded.

Facial Movements and Their Role in Animal Research

Fine motor behavior in the face has been primarily studied in the context of affective neuroscience and psychology, mainly in humans[37, 51]. Studying facial movements in other species can not rely on many of the assumptions being made there, for example that they are used to communicate or that they reveal information about an individual's internal state. The latter assumption is still under active investigation and lacks final verification[2, 5, 95]. This poses the issue of how to extract facial movements from animals without prior bias or anthropomorphic assumptions so that later correlation with recorded neural activity is as little skewed as possible to the initial hypothesis. So far, many studies proposing systems of analysis of facial movements in other species failed to acknowledge the particular conditions that differ from work in humans[68, 99, 113].

Anatomy of the Cortico-Bulbar Tract

Control of facial musculature is exercised through the cortico-bulbar tract that links primary M1 to the facial nucleus of the brainstem which connects to cranial nerve VII, the facial nerve (FN). The FN exits the skull through the facial canal and synapses to muscles of the face. It controls facial motor actions (including the corneal reflex), the difastric muscle (under the jaw), stylohyoid muscle (side of the jaw) and the stapedius muscle (bones of the inner ear). It also receives sensory input from the anterior two thirds of the tongue as well as innervating submandibular, lacrimal, sublingual glands and the nasal mucosa. [88]

Other than the FN, cranial nerves V (trigeminal nerve), IX (glossopharyngeal nerve) and XII (hypoglossal nerve) are efferents of the cortico-bulbar tract. Damage or dysfunction of the cortico-bulbar FN connectivity and function thus mainly affects the motor control of the face. From general palsy (Bell's palsy), that can also be caused by viral infection [48], to a wide range of syndromes (Ramsay-Hunt, Guillain-Barre, Heerfordt, Melkersson-Rosenthal, Carey-Finemann-Ziter, Moebius-Syndrome) or the symptoms of Neuroborreliosis or stroke (supranuclear lesion) many conditions impair the function of the FN. This can have significant impact on the quality of life, since facial expression is such an important factor in human social interaction.

Studies on the specific workings of the cortico-bulbar pathway are very limited, when compared to the extensive research on fine motor control exercised through the cortico-spinal tract [91]. Studies of cortico-bulbar pathways in rodents often focus on licking (cranial nerve XII, hypoglossal nerve) [14, 77] and whisking [65, 82] since they are major rhythmic motor actions that are easy to quantify. A recent review by McElvain et al [80] summarizes the current state of knowledge concerning those orofacial behaviors and their anatomical foundation. Those studies mainly link sensory systems and motor behavior that are both coupled to central pattern generators (CPGs), above all the pre-Bötzinger complex that regulates the breathing rhythm in the context of sampling the environment. In recent years, rodent vibrissal motor cortex has been established as a model circuit to study the interactions of M1 with CPGs and muscles directly [22]. Vibrissal motor cortex is known by many abbreviations [38]. In the following the abbreviation vMCx is used.

Aim of this thesis: Computational Analysis of Facial Motor Actions

Very few studies have focussed on the concerted fine movement of different parts of the face [37, 101]. As of yet none have been investigating the role of M1 in the encoding

of multidimensional, concerted facial motor behavior. On the other hand, concerted facial motor action has been the subject of intensive study in humans, both in the fields of Psychology, Affective Neuroscience and Computer Vision [17]. Building on findings from these diverse fields we aimed at characterizing the relationship between neuronal activity in the facial area of the M1 and spontaneous face movements. In the following I will introduce how this thesis provides the computational toolbox to automatically extract a metric of facial movements in head-fixed mice.

2. Facial Nerve Circuitry

2.1. The Facial Nerve

The facial nerve is the 7th of 12 cranial nerves (CN VII) FIGURE 2.1. Its origin is the pons of the brainstem connecting to motor functionality and the intermediate nerve connecting to sensory functionality, respectively. Those two nerve strands join and pass through the internal auditory meatus into the petrous part of the temporal bone. It then takes three different paths transversing the face canal and finally connects to the geniculate ganglion, the chorda tympani (which primarily transmits information from the tastebuds of the anterior two-thirds of the tongue and the parasympathetic connections to sublingual and submandibular glands), nucleus petrosus major (eventually controlling the lacrimal gland), and parotid plexus (connecting to muscles of the face).

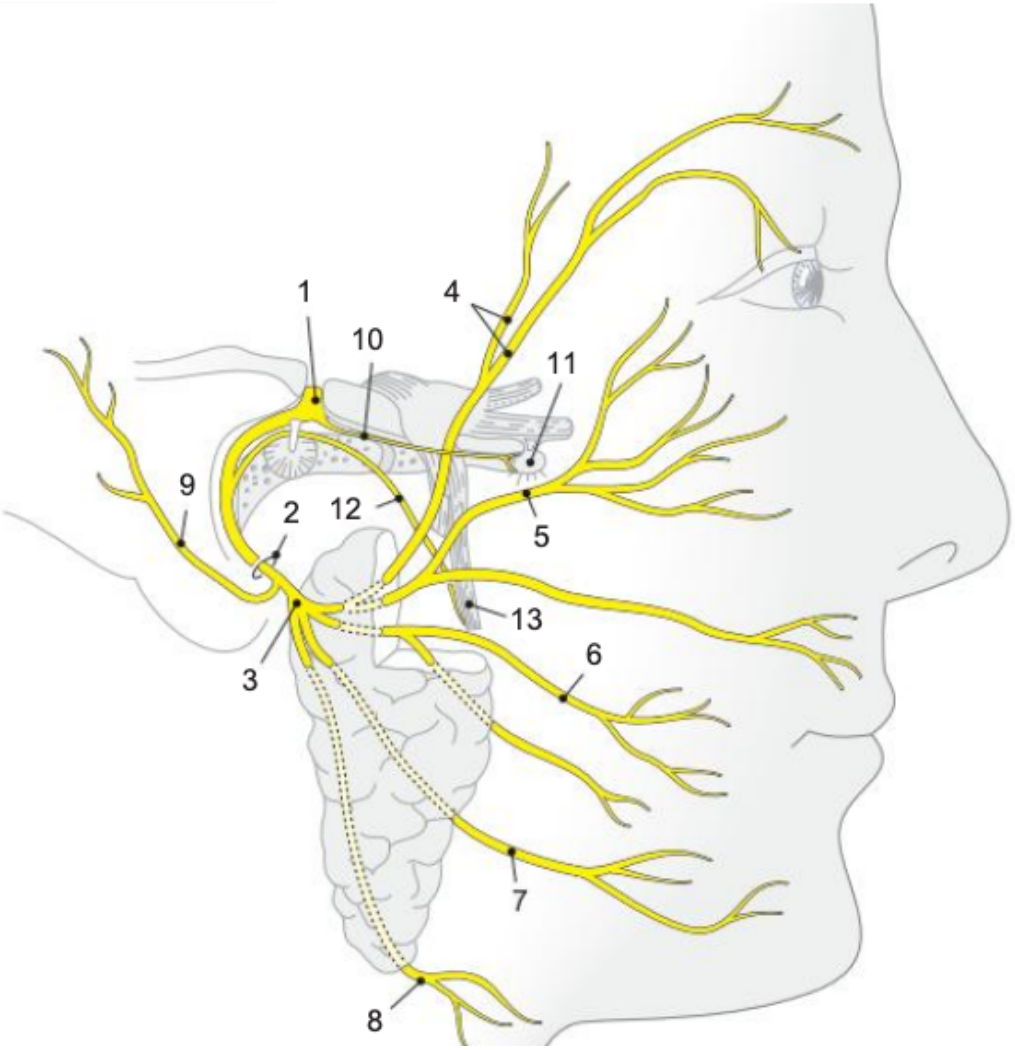

FIGURE 2.1.: Anatomy of the facial nerve. 1: geniculate ganglion; 2: Foramen stylomastoideum; 3: Parotid Plexus intersection; 4-9: Nerves to face muscles; 10: Nucleus petrosus major; 11: ganglion pterygopalatinum; 12: Chorda tympani; 13: Nucleus lingualis

2.2. Connections of M1 to the Facial Nerve

Most fibers that control the muscles of the face cross the midline, but some establish ipsilateral connections to the motor nucleus. [72] The biggest difference in between humans and rodents is the vibrissal system that enables tactile sampling of the environment by precise motor control of the whiskers. Motor control of the whiskers is exercised through extrinsic and intrinsic muscles [21, 118]. Both of these muscle groups are mainly controlled by the FN.

The non-direct connections of M1 to the FN have been the subject of many anatomical studies [22, 38, 80]. The increase in interest in the connectivity of vMCx is also caused by the close integration of sensory areas into the circuitry, making it a good model circuit to study sensorimotor integration. Early on, functional connectivity between vMCx and whisking behavior has been established [13, 100].

Anatomical connectivity of vMCx has been studied by multiple tracing studies. Its main targets were determined to be the FN nuclei in the brainstem and the reticular formation (RtF). Some sparse direct monosynaptic connections from vMCx to FN were found [53, 100]. FN itself has been mapped extensively in rats. It is described as an ellipsoid-shaped nucleus 1.3 ± 0.03 mm lateral-medial, 1.6 ± 0.02 mm rostral-caudal and 0.9 ± 0.06 mm ventral-dorsal of bregma [54].

Whether or not the trigeminal nuclear complex (PrV) is also a target of vMCx is the topic of some debate [81, 96, 100]. Smith et al and others observed M1 projections to mesencephalic and brainstem structures such as the superior colliculus (SC), the periaqueductal gray, the basal pons, the deep mesencephalic nucleus, the interstitial nucleus of the medial longitudinal fasciculus and the gigantocellular, parvocellular and intermediate reticular nuclei [96] Trigeminal pathways project to SC and vibrissa intermediate reticular formation (vIRt), the latter one being the pre-motor whisking oscillator. Both formations are in turn are connected to CPGs in the brainstem [36] Figure 2.2.

2.3. Central Pattern Generators in the Brainstem and Their Importance in Face Motor Activity

Those CPGs are hypothesized to be important regulators of rhythmic activity such as sniffing, whisking and licking, all of which are major contributions of the total face motor activity displayed by rodents. The most important rhythmic activity linked to all those behaviors is breathing. Breathing frequency in rodents was reported to be 7 Hz in

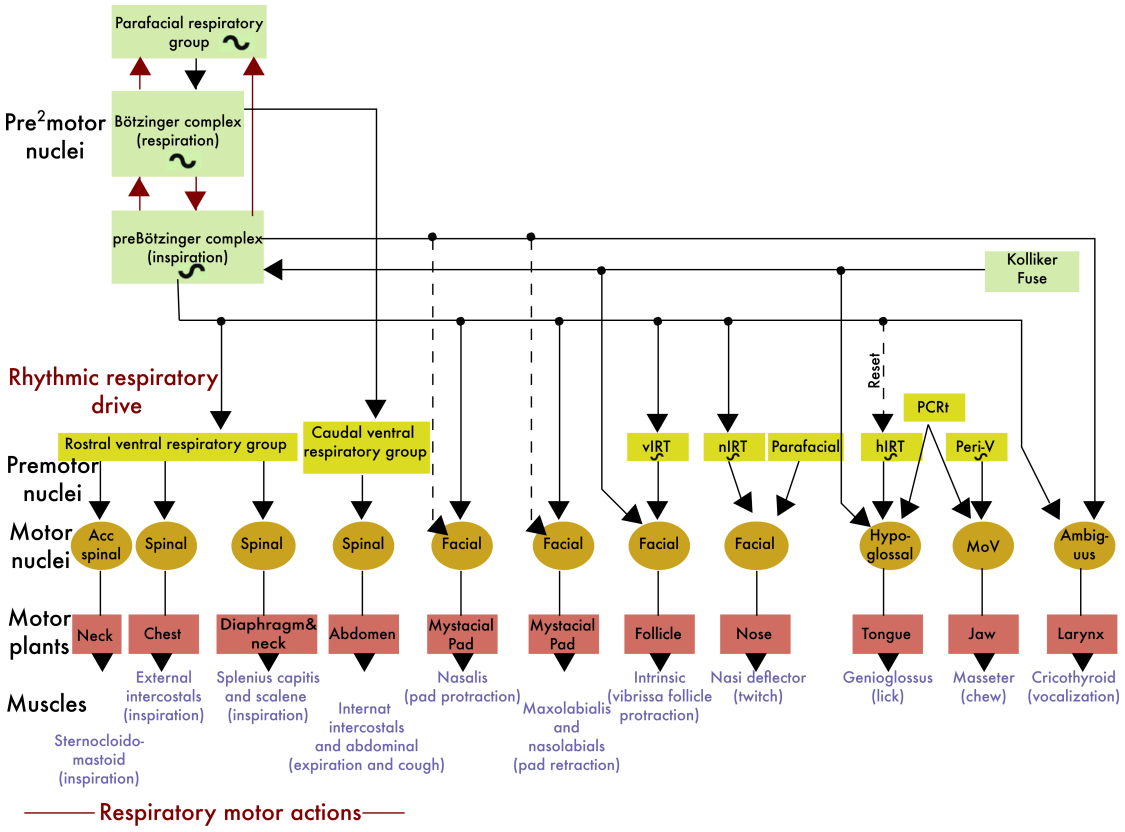

FIGURE 2.2.: CPGs in the brainstem influence a wide range of motor nuclei. Most motor behaviors are influenced this way. The putative neuronal oscillators are marked with a ~. Breathing control centers (green) project to putative premotor controllers of diverse orofacial musculature (yellow). Dashed lines are connections based on functional rather than anatomical data. Abbreviations: Acc. spinal (accessory spinal nucleus); MoV (motor trigeminal nucleus); nIRt, hIRt, and vIRt (nasal, hypoglossal, and vibrissa intermediate reticular formation, respectively); PCRt (parvocellular reticular formation); Peri-V (peri-trigeminal area).

rats and 11 Hz in mice on average [80]. Respiration has been found to be initiated by the pre-Bötzinger complex, that serves as the principal oscillator. It directly innervates iRt and vIRt formations. vRt plays an important role in whisking since it sends rhythmic inhibitions to the muscles of the whisker pad [65, 82]. Thereby it establishes the phase locking of whisking to breathing so that every initiation of a breath can be regarded as the initiation of a whisking epoch [67]. Although definite experimental proof still stands out, most likely the pre-Bötzinger complex also regulates the rhythmic movements of nose, tongue and even the head.

2.4. Tongue Movement is the Second Most Studied Face Motor Behavior

Another face motor behavior that is at least in part linked to the pre-Bötzinger complex is tongue movement. The tongue is controlled by the hypoglossal nerve (CN XII), which also linked to M1 through the cortico-bulbar tract. The tongue is moved by a set of extrinsic and intrinsic muscles [1, 73, 98]. Those muscles are linked to the hypoglossal motor nucleus (CN XII), where integration with sensory feedback from trigeminal and vagus nerve afferents is taking place [19, 120]. Since licking is phase-locked to breathing as well [115], integration to CPGs is also happening at this level. The current hypothesis is that the hypoglossal oscillator is found in the IRt, too (hIRt) [116]. In the midbrain, SC is linked to the tongue CPGs. On the cortical level, two M1 areas have been linked to tongue muscles in mice; posterior-medial M1 has been identified by retrograde labelling (anatomical connectivity) and anterior-lateral M1 (ALM) has been identified by microstimulation studies (functional connectivity) [66]. ALM has also been identified as an important area in voluntary licking under sensorimotor learning tasks [55, 56, 69, 70]. However, M1 was shown not to be essential in feeding-relating licking [119].

Komiyama et al shed light on the interesting involvement of M1 in learning. They found that temporal correlation of neuronal activity depended on two factors: Spatial proximity and, more importantly, their response type. Correlation of neurons with the same response type increased after learning the behavior. Plasticity in M1 controlling motor behaviors through the cortico-bulbar tract that decreases the variance in population coding remains to be investigated on a wider range of orofacial behaviors [66].

2.5. Summary of Issues When Studying M1 Control of Face Movements

The complex integration of sensory and motor circuitry at the level of CPGs in the brainstem shows that assumptions concerning direct control of face motor behavior by M1 can not be made easily. An interesting question is to what degree the direct M1 to whisker pad connections influence this behavior. Can those neurons with monosynaptic connections to the FN motor neurons [53, 100] be identified from population recordings? Is M1 mainly an initiator of rhythmic activity that is controlled by FN and CPGs in the hindbrain? How much does activity at motor cortex even correlate with face movements? Can learning, that is fine-tuning of automated movement patterns, be observed at the level of cortical activity?

3. Facial Motor Actions

3.1. The Anthropomorphism of Facial Expressions

Fine motor control of facial movements in humans play an important role in conveying complex information as a mean of non-verbal communication. In this context it is more common to speak of *facial expressions*. While it is not debated whether or not facial expressions add a layer of meaning to our communication, this question remains unanswered for other species [2, 3, 6]. Since Charles Darwin published his book *The expression of the emotions in man and animals* in 1872 [32], the study of facial conformations and what they express about the interior mental state of an individual has been extended from humans to many other animal species [35, 113]. In his book, DARWIN introduced the hypothesis that

> The community of certain expressions in distinct though allied species, as in the movements of the same facial muscles during laughter by man and by various monkeys, is rendered somewhat more intelligible, if we believe in their descent from a common progenitor.

According to this, different species not only share similar biomechanical properties of face muscles thanks to a common progenitor, but also the mental process causing their coordinated contraction, in this case laughter. The latter idea has been extended by others [41, 42, 104] and today it's known as the basic emotion theory [40, 62].

3.2. The Basic Emotion Theory

The basic emotion theory states that all humans share an essential set of basic emotions, found in all cultures and individuals. As a consequence the expression of these emotions, particularly on the face is thought to share the same essential features as well and can thus be interpreted universally. Its wider definition thus also applies across species, primarily mammals.

In human Affective Psychology, FACS established by Paul Ekman and Wallace Friesen in the 1970s is one of the standard approaches to infer an internal emotional state

of the test subject based on its facial expression. It is based on the concept of action units (AUs), that is a set of observed conformations of all 43 muscles of the human face making up every expression displayed by it. The original publication (1978, Consulting Psychologists Press) is not accessible digitally, even though the Library of Basel still holds some original hard copies. The revised FACS was published as a non-peer reviewed commercial manual in 2002.

FACS nowadays is the theoretical basis for similar approaches in other species [15, 16, 35, 46, 87, 108, 110, 111, 113]. It has to be noted that the push for the widespread use of FACS for application in different animal species seems to come from a relatively limited circle of researchers (see those citations).

3.3. Current Approaches in Face Movement Classification for Animals

Those essential assumptions of the basic emotion theory, and by proxy the assumptions on the universality of facial expressions, have become the topic of strong debate among psychologists [5, 20, 50]. Despite that, up to recently many ethological studies still adapted those postulates unquestioned. While studies in human can rely on some sort of ground truth in the form of questionnaires and interviews, animal experimenters essentially need to assume that Darwin's theory of the homology of emotions and their expression across species holds true. This could not only confound the results of studies researching non-verbal communication within a species other than humans, but is particularly difficult when human researchers have to evaluate the motor output of animals. Notably this is the case in fields such as Pain Research or Behavioral Neuroscience where it is crucial for animal welfare as well as the analysis of behavioral data to determine the affective internal status of an animal as quickly and reliably as possible. For applications in Pain Research in rodents the Mouse Grimace Scale (MGS) and the Rat Grimace Scale (RGS) [68, 99] have been proposed. Both scales were created by showing pictures of rodents after application of varying amounts of a pain-inducing agent to "human facial pain expression experts" blind to the true condition of the animal and letting them score what amount of pain they believed the animal to be in. The experts were asked to make their judgement based on the conformation of 5 areas of the mouse face, similar to AUs in FACS (those being here: orbital area, surrounding the face – orbital tightening, back of the nose – nose bulge, cheek - cheek bulge, ears – ear position and whiskers – whisker position change). Those human coders reached accuracies varying from 72 to 81%. In rats a slightly altered *pain face* was observed,

resulting in small alterations of the definition of the action units of the MGS for use in rats. The ground truth of a pain-causing agent per se is known, but the authors avoid discussing the possibility of confounding anthropomorphism in the design of the scale concerning arousal. The rodent grimace scales are still the method of choice for Pain Research in rodents, showing the lack of an unbiased pain scoring system based on pain-related brain activity ground truth. An algorithm detecting stereotyped fine motor output in the mouse face could solve the scoring part of this problem.

3.4. Computational Advances

The general task of automating the analysis of video recordings of animals has gained a lot of attention recently [76]. While many approaches rely on deeplearning-based automated overall pose estimation of the animal [39, 52, 75, 85, 90, 121], the more specialized field of automated analysis of face videography in animals has been explored less. One example is the library FaceMap [101] and very recently an approach to classify movement in the mouse face by DOLENSEK et al [37]. MUSALL et al used a metric derived by simply averaging pixel value differences of different areas of the mouse face throughout the video [84].

On the other hand, facial recognition, especially facial expression recognition has attracted considerable interest in Computer Vision research. But those algorithms are almost exclusively designed to work on human faces. Scale and important features of the mouse face differ a lot from human faces: The mouse skull has roughly 1/6 of the size of the human skull, so that the scale of detected features needs to be altered at least. But also the nature of the features differs significantly: Mouse skin is covered by dense fur, its most prominent moving parts are whiskers, nose and ears. This means that one can not rely on AUs or similar metrics established for the comparably flat, bare-skinned human face.

4. Facial Expression Recognition

4.1. Dimensionality Reduction of Videography Data

Recently, developments in the fields of Computer Vision (especially object detection and feature tracking) and Artificial Intelligence (AI) advanced so far that they are now also applied to problems concerned with facial expressions. The detection of human faces is a common task for automated surveillance, digital photography and human-to-computer interfaces. Since the first real-time object detection framework was introduced by Paul Viola and Michael Jones in 2001 [109], a vast number of algorithms has been proposed to automatically detect human faces in video recordings, rendering it an essentially solved problem [17]. However, detection of features within a human face and tracking them is a more difficult task. Several types of features have been successfully applied to it, such as Haar [109], Histograms of Oriented Gradients (HOG) [117], Scale-Invariant Feature Transform (SIFT) [73] and simple edge detection [18]. Those approaches extract *first-order features*, those being features that can be extracted from the base image. With the recent advent of fast deep neuronal networks (NNs) and convolutional neuronal networks (CNNs) many solutions now rely on the NN tracking features that only become apparent after multiple layers of convoluting different kernels over the image data, *nth-order features*, which opens the possibility to track features that are not readily visible to a human observer.

All of these methods essentially aim at extracting a low-dimensional feature representation of the high-dimensional video frames. The time dimension poses the bigger issue: How can movements/features be tracked reliably over the chronological time course of a video recording? The issue here is that the same feature needs to be identified across frames.

4.2. Temporal Component of Movement

Estimating movements of objects over multiple frames of a video recording is a problem tackled in the Computer Vision field by optical flow algorithms[7]. They attempt to

interpret the projection of 3D movement onto the 2D imaging plane as a vector field. Variables described by the vector field are either the estimated pixel displacements or "the velocities of movement of brightness patterns in an image" ([73]). Crucially, optical flow is distinguished from object and feature detection methods described above by explicitly including the information contained in the time dimension.

Alternatively, instead of analyzing the differences in between consecutive frames (that is analyzing movement), it is possible to compare the individual still frames themselves (that is analyzing face conformations). Ideally low-dimensional representations of the single frames, obtained by one of the methods mentioned can be used in the calculation of pairwise distance or similarity (similarity = 1 - distance) measures. The resulting distance/similarity matrix can then be clustered, yielding a scalar descriptor of every single frame. A sequence of these descriptors (the `classification vector`) is essentially a low-dimensional descriptor of the movement across those frames.

4.3. Extracting Stereotyped Movements

The notion that most animal motor behavior can be classified into a small set of repetitive stereotypic bouts of movements is the basis for most analyses in behavioral sciences[4, 8]. Today, automated data recording and classification by NNs makes this way of studying animal behavior relatively easy and fast.

A crucial step is the training of this network, since the majority current approaches rely on supervised learning[112]. This is the step in which human bias is introduced in the analysis, since the network just learns to classify the data like the human experimenter that trained it. Recent approaches try to circumvent this training error by analyzing the structure of the raw movement data itself[83]. For example, automated stereotyping of movement bouts has been proposed in Drosophila[9]. BERMAN et al proposed to apply Morlet Wavelet transformation to time series data of motor behavior. Morlet Wavelets are originally used in applications in Geology, where they are used to isolate periodic seismographic activity from noisy seismographic recordings[25]. Similarly, dimensionality-reduced time series data of motor behavior contains the periodic bouts of stereotypic movements hidden in the noise of the recording. A simpler approach has been proposed by STRINGER et al [101]. Here a low-dimensional representation of the videography data was obtained by simply applying singular value decomposition (SVD) to the three-dimensional tensor of the video data. They avoided extracting stereotypy from this output, and instead correlated it with neural population activity directly. In theory, different approaches could be used to analyze periodicity and stereotypy of

this continuous time signal, including wavelet transforms. We tried several of these approaches *see* CHAPTER 7.

Finally, DOLENSEK et al instead generated a discrete output of the video signal by applying a trained classifier on every individual frame[37]. Their approach combines the `classification vector` representation described above with supervised learning of a random forest classifier.

5. Aim

The aim of this thesis is to develop a library, written in MATLAB, that automatically and unbiasedly extracts stereotyped face movements from two-dimensional videography of head-fixed mice. Specifically, the face behavior is spontaneous, meaning that no external stimulus is provided to the mice in order to cause a certain movement. This library is intended to be used in the wider context of Juan Carlos Boffi's experiment to characterize the relationship between M1 and spontaneous face movement in head-fixed mice. The results will present the final version of that library. The commented source code can be found in FUNCTION A.4.

Part II.

Results

6. FaceCat: A Library to Extract Stereotyped Face Motor Activity

This chapter will describe the logic of the final version of the library (termed FaceCat) to extract stereotyped face movement in head-fixed mice. For in-detail explanation of the source code *see* FUNCTION A.4. The mathematics of HOGs are explained in detail in FIGURE D. Briefly, FaceCat each video frame into a vector, before calculating the distance between these vectors. The resulting distance matrix is hierarchically clustered, obtaining a `classification vector` with one scalar entry corresponding to each frame. Last, stereotypic sequences of these scalars are detected.

6.1. Classification of Still Frames

The steps of the two central processing functions are outlined in FUNCTION A.4. First, the user needs to specify the full paths to the video files that are to be analyzed. The first frame of every video is then opened in an interactive window that allows the user to specify the regions of interest (ROIs) for eye and snout per drag-and-drop FIGURE 6.1.

FaceCat was designed around a two-photon Ca^{2+}-imaging setup FIGURE 6.2. This means the mouse is head-fixed and constrained to a plexiglass tube, resulting in minimal movement of the filmed face. The camera was placed 5-10 cm in an orthogonal position to one side of the mouse face. The sizes of those ROIs are preset to ensure repeatability and comparability of analyses of different videos. This first step also ensures that the correct regions are subject of the analysis, since there might be small differences in camera position and angle. The sections of the video during which the two-photon IR laser is scanning the brain are automatically determined by calculating the average grayscale value of the eye-ROI FIGURE 6.1. During recording epochs the eye appears white on the IR-sensitive sensor of the camera. The indices of the recording epochs are then stored internally to perform subsequent analysis steps only for those periods of interest FIGURE 6.3(a).

Histograms of oriented gradients (HOGs), for the in-depth description of the proce-

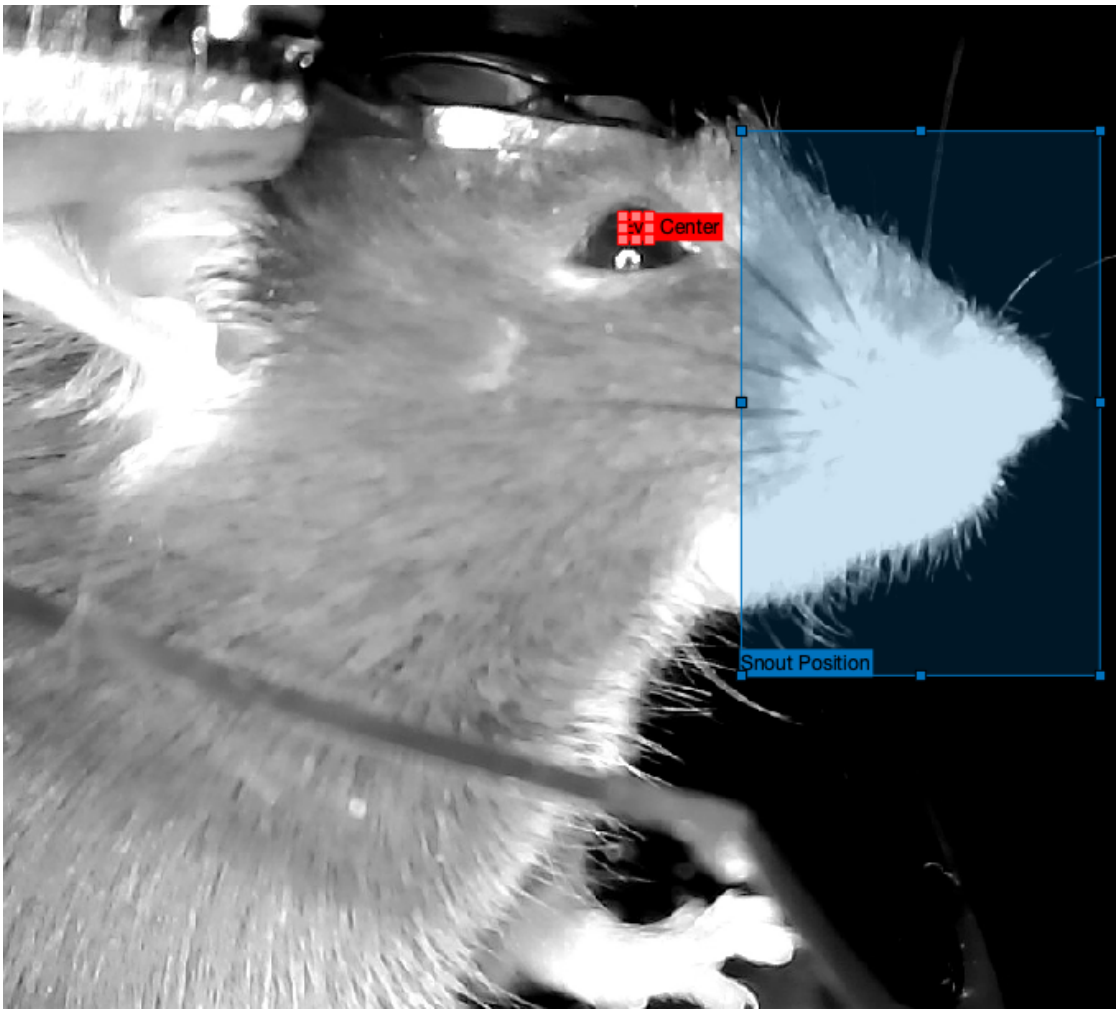

FIGURE 6.1.: Selection of face region to analyze. ROI of the eye center (red) used to determine if the IR laser was on during the frame. ROI of snout area (blue) used for subsequent analysis steps.

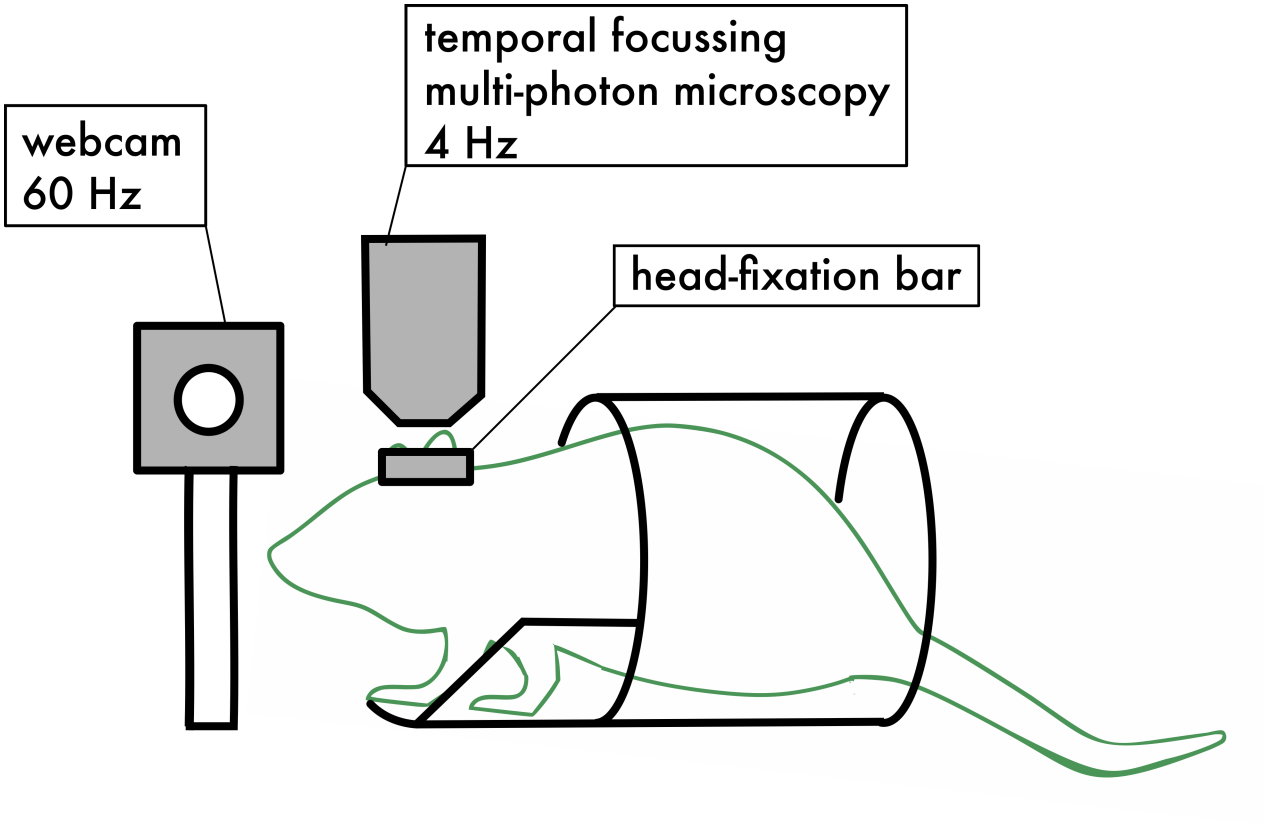

FIGURE 6.2.: Sketch of the experimental recording setup. FaceCat is designed around recording one side the face of a head-fixed mouse that is constrained to a plexiglass tube. Camera records at 720p resolution at 60 Hz.

dure *see* FIGURE D, are calculated as described in [37] FIGURE 6.3(b). The operation is only performed on the snout-ROI region of the video, as specified by the user. The HOG vector for every frame is stored. Calculation of all pairwise cosine distances between those HOGs yields a distance matrix that indicates similarities of the facial conformations displayed in every frame FIGURE 6.3(c).

This distance matrix is already suited to determine clusters of similar face conformations. To determine the number of clusters, the multidimensional HOG vectors are embedded into a 2D- space using t-SNE FIGURE 6.3(d). The discrete scatter plot is converted into a continuous pdf by convolution with a Gaussian kernel, $\sigma = 1.8$ FIGURE 6.3(e). This part of the code is adapted from [9]. The number of clusters is then estimated by calculating a watershed transform, that is by delineating regions of the surface that share the same local maximum FIGURE 6.3(f).

The number of regions determined this way is used as the number of clusters when performing hierarchical clustering on the distance matrix FIGURE 6.3(g). The result of this analysis step is a vector of a length equal to the number of frames of the video recording FIGURE 6.3(h).

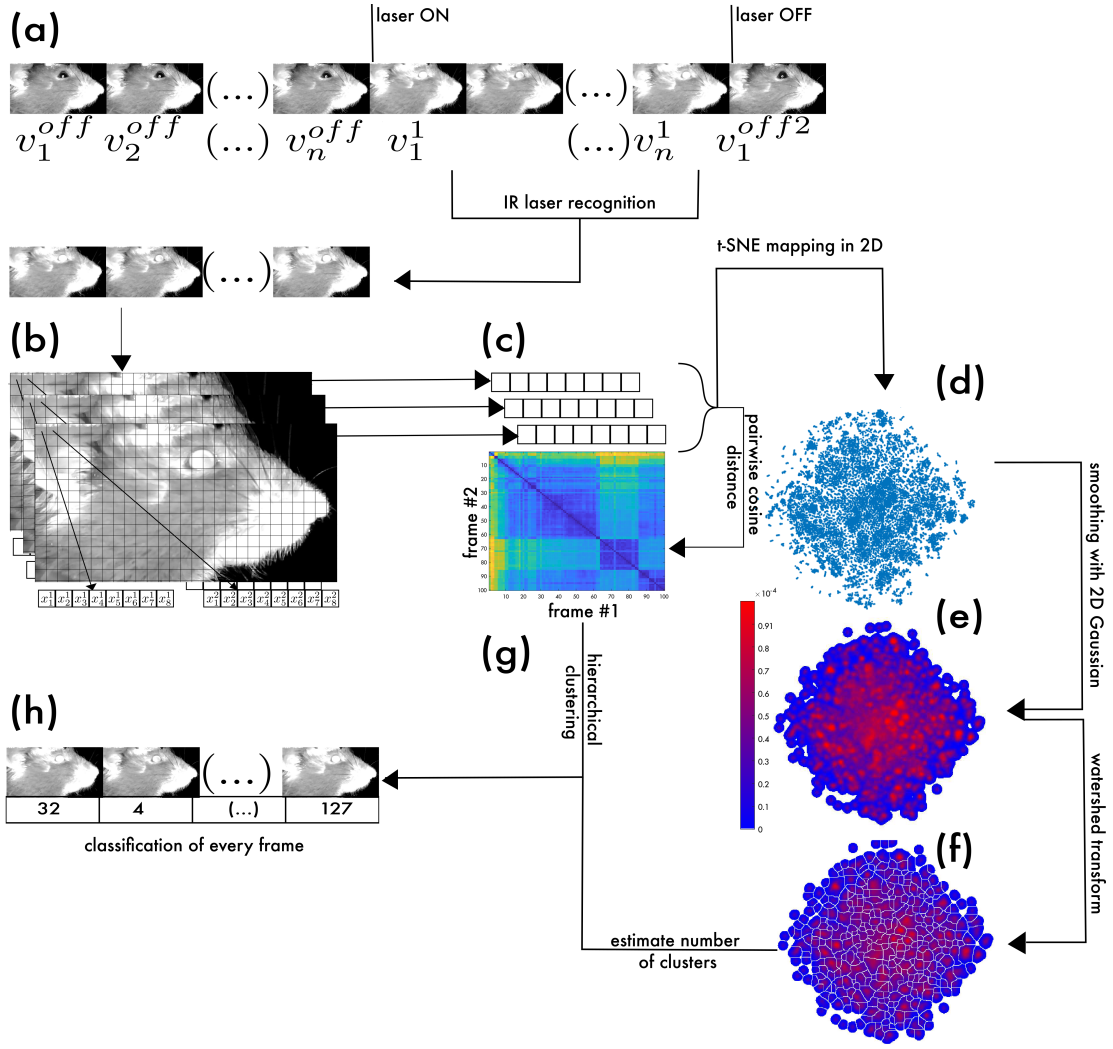

FIGURE 6.3.: Schematic outline of the FaceCat algorithm. (a) Automated recognition of IR laser recording epochs based on grayscale value of the eye. Input is a series of still frames v, superscripts denote recording epoch number, subscripts the frame number within that epoch. (b) HOG vector extraction; every still frame is converted in the same number of HOGs that are then concatenated in one HOG vector. (c) Pairwise cosine distance calculation of all HOG vectors. (d) t-SNE embedding of all HOG vectors in a two-dimensional space. (e) The discrete t-SNE plot is convolved with a Gaussian kernel (σ = 1.8) in every dimension. The result is a probability density (f) Watershed transform of the probability density. The boundaries of the delineated areas are indicated as white lines. The number of discrete areas equals the number of clusters detected. (g) The estimate of the number of clusters is used to hierarchically cluster the distance matrix. (h) Classified still frames. Typically around 130 different clusters were identified.

6.2. Analyzing Stereotypy

This classification vector is then used for the final analysis steps. The aim is to determine which sequences of face conformations are repeated, how many times they are repeated and whether or not slight variations of the same sequence pattern exist. The algorithm offers two options at this point: Either treat the vector as a discrete sequence of n patterns, where n is the number of frames divided by the set pattern duration Δt FIGURE 6.4(a), or to use a sliding-window type approach, where every single pattern of duration Δt starting at every frame of the video is being analyzed FIGURE 6.4(b). Either way, the patterns of duration Δt are extracted from the classification vector and stored in a matrix. Duplicates and circular shift permutations are removed at this point. Then, all patterns are convolved with the classification vector, counting the number of equal values at every step FIGURE 6.4(c). The resulting overlap count vector length is Δt - 1 entries shorter than the classification vector. Since this is done for all previously detected patterns, one overlap count vector per pattern is obtained. The overlap count is then normalized by dividing every single entry by Δt which gives a relative overlap score between 0 and 1 (1 being total overlap) at every timepoint considered FIGURE 6.4(d).

Statistics that are then performed on the overlap score matrix include counts of the number of repetitions of every pattern (equal to the number of values equal to 1 in the corresponding row of the overlap score matrix), their locations and counts and locations of patterns similar to that one. Most importantly, a similarity measure of the patterns is established FIGURE 6.5(a); a pairwise overlap count comparison of two randomly selected patterns with each other is performed 10,000 times. This approach, also known as bootstrapping, yields a distribution of similarity values that is a good estimator for the hidden distribution of possible similarities of patterns with each other. The 95% quantile of this distribution was chosen to be the threshold for similarity FIGURE 6.5(b). Any two patterns whose pairwise similarity is higher than that value is *similar* for the purposes of the following analysis steps FIGURE 6.5(c). This means stereotypy as well as similarity of patterns can be determined.

Finally, the algorithm stores all results in a structure array TABLE 6.1. Here, all unique patterns are stored in a list together with the indices of the locations of that pattern within the classification vector and the cutoff values derived from the bootstrap calculation. The distance matrix of the unique patterns and the classification vector are also contained for future reference.

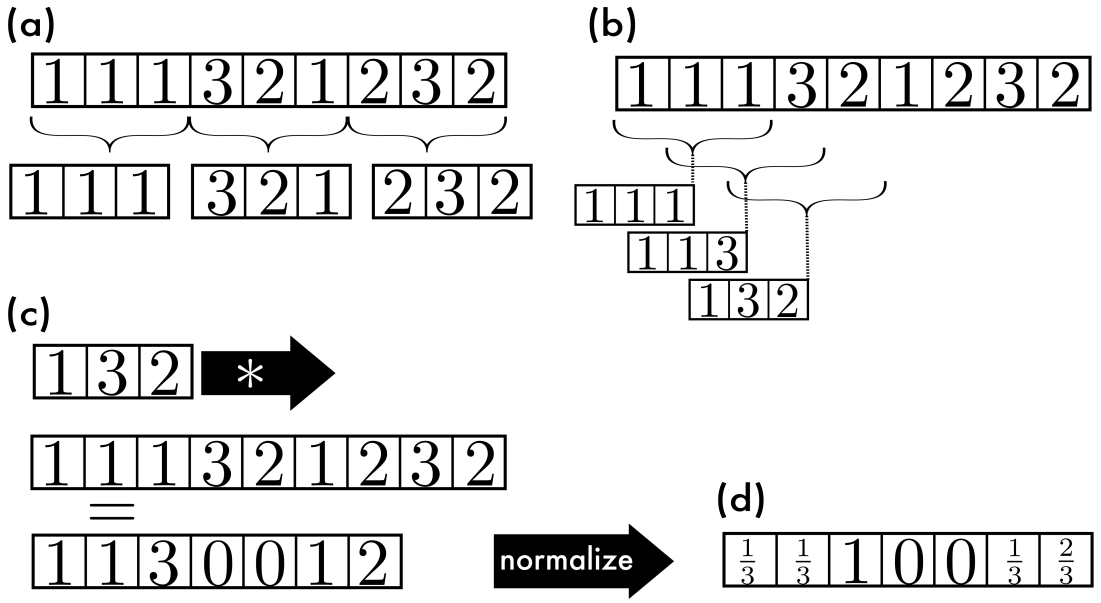

FIGURE 6.4.: Schematic of pattern extraction and overlap count procedure. (a) Discrete extraction of patterns from the classification vector. Patterns have the exemplary length (Δt) of 3 frames. (b) Sliding window extraction of patterns. (c) Convolution of patterns, here exemplary pattern #3 with the classification vector. The similarity at every step is determined by counting the number of identical entries (overlap count). No zero-padding. (d) Normalization of the overlap count (score).

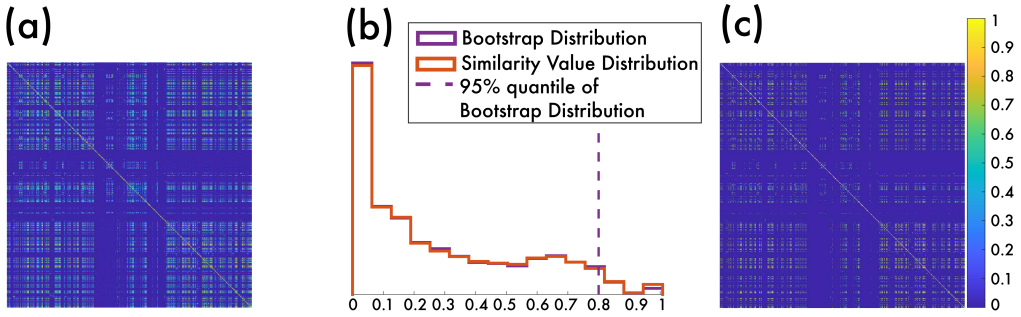

FIGURE 6.5.: An exemplary overlap count matrix (similarity matrix) of all patterns of a recording before and after thresholding. (a) The pairwise similarity matrix of all unique patterns. (b) Comparison of distributions of random similarity vs observed reality. y-axis shows relative units. (c) Similarity matrix after setting all values below the 95% cutoff to 0.

Output Structure: Pattern Analysis					
List of Patterns	Overlap Score	Overlap Locations	Classification Vector	Distance Matrix	Cutoff Values

TABLE 6.1.: Variables contained in the output structure.

6.3. Practical Application

FaceCat was primarily designed as part of the analysis pipeline of Juan Carlos Boffi's experiment concerning the relationship of M1 with face motor behavior. In the next analysis steps, he defined four categories to classify the patterns:

1. Recurrent Invariant patterns
    - Stereotyped, repetitive, trained or learned motor programs.
    - Recurrent Invariant patterns are exactly the same, every time they are recorded. This means the sequence of n facial conformations during the timespan Δt never changes
2. Recurrent Variant patterns:
    - Adaptations or readjustments, late stages of learning.
    - Recurrent Variant patterns occur multiple times throughout the video recording, but they show slight variations every time they happen. At the same time they are similar enough (see definition for the threshold above) to not be classified as entirely different patterns
3. Non recurrent Variant:
    - Highly Adaptive, motor patterns, early stages of learning.
    - Non Recurrent Variant patterns are patterns that are similar to each other (see definition for the threshold above), but individually they are only observed one single time per experiment recording session
4. Unique patterns:
    - Unique patterns happen only once and do not bear any similarity to any other activity pattern

These categories are used to determine whether a behavior pattern has been learned over time, in which case the variability of closely related patterns should have decreased.

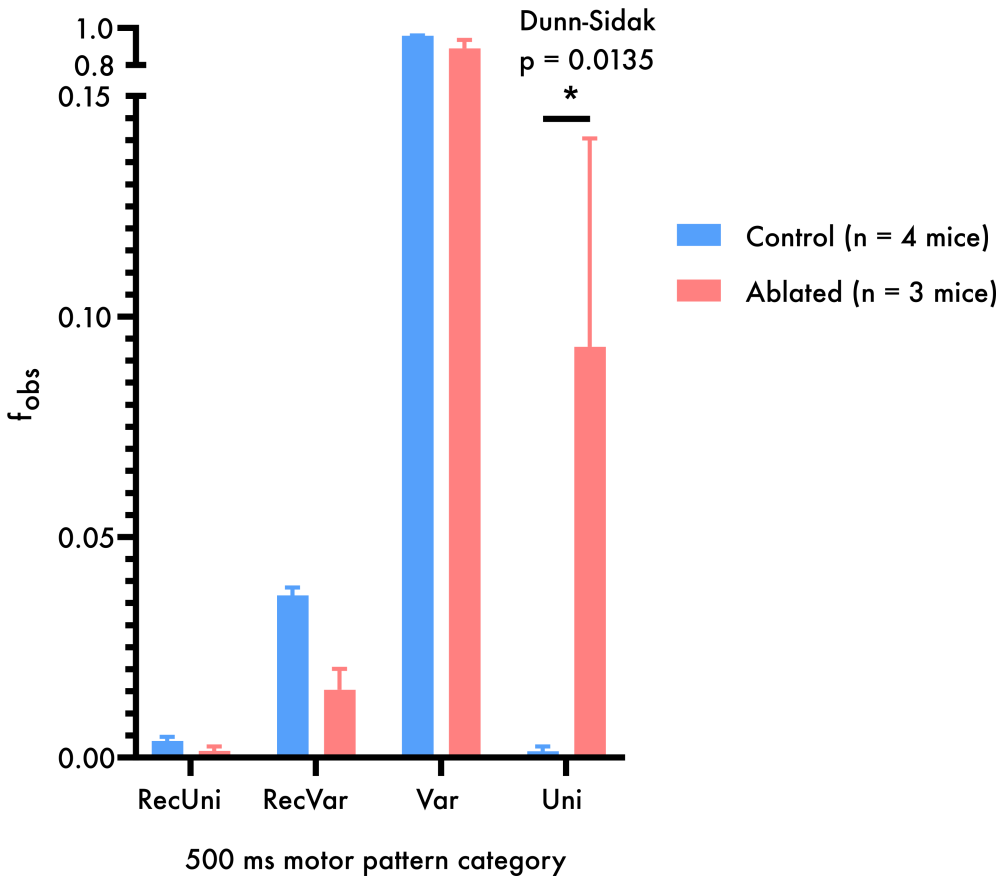

FIGURE 6.6.: Primary results when quantifying face movement patterns in one of the four categories recurrent invariant (RecUni), recurrent variant (RecVar), variant (Var) and unique (Uni). Duration of patterns $\Delta t = 500$ ms. Quantified is the relative observed frequency (f_obs).

He applied the algorithm to head-fixed mice with intact M1 (n = 4) and to a separate set of mice after surgical removal of the face area of the motor cortex (n = 3). Primary analysis was performed extracting patterns of length $\Delta t = 30$ frames (500 ms) using a sliding-window (*see* FIGURE 6.4 (b)). Statistical quantification shows a significant increase (*Dunn-Sidak test*, $p = 0.0135$) in unique patterns after ablation of M1 FIGURE 6.6. This parallels findings in the cortico-spinal tract that M1 is vital for dexterous motor behavior[49]. There is a non-significant tendency of a decrease in the number of recurrent invariant, recurrent variant and variant patterns. This observation corresponds with prior findings that ablation does not influence previously learned behaviors[49, 91].

In summary, the algorithm not only outputs a classification vector that can be cor-

related with the simultaneously recorded Ca^{2+}-imaging activity patterns, but can also give information about stereotypy and similarity of facial motor action patterns. This is being tested in ongoing experiments, whose primary results indicate that FaceCat, as part of the wider experimental design, is suited to quantitatively study dexterous motor control of on-going, untrained, spontaneous behaviors.

7. Alternative Approaches

This sections covers the first approaches we developed to extract a metric of face movements. They all were dropped eventually, but the validity of the general algorithms is outlined and the reason why we opted for another option discussed briefly.

7.1. Classification of Still Frames

The first approach taken aimed at determining a number of pixels of interest from the videography data and to track them reliably. We hypothesized that correctly defined pixels of interest should be able to serve as descriptors of the overall face movement.

7.1.1. Markerless Pose Estimation of Face Landmarks

Deeplabcut (DLC) [75] is a user-friendly implementation of DeeperCut [61] written in python. It is a deep neuronal network based on the ResNet architecture [58]. Features provided are fully automatic point feature tracking in behavioral video recordings of animals that has been used in a wide range of experimental applications. DLC abolishes the need for physical markers on the animal in order to track those points as well as the manual analysis step of the recorded video. It relies on supervised learning however, so that the user has to provide manually labelled input of a small subset of the recording. The authors claim that as few as 200 frames are enough to train the network to satisfying accuracy. The bottleneck of any supervised learning approach is providing this pre-classified training data as a ground truth for the neural network to train itself on. It presents a twofold challenge: First, the time needed to be invested into manually creating the pre-classified training data and second the challenge of generating unbiased training data. The second concern is the more important one, because it is almost impossible for a human to reliably detect the important components, usually referred to as features, of movements in the mouse face. While humans can rely on a lifetime of experience in classifying and interpreting other human faces, finding the critical features in an animal's face is a task we are not trained to do. Providing manually generated ground-truth to the neural network thus might just introduce the kind of bias

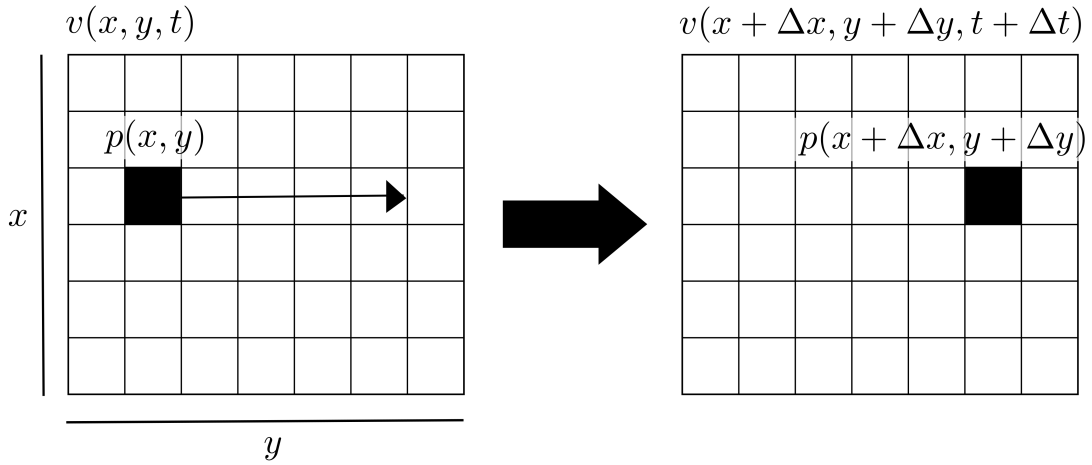

FIGURE 7.1.: Optical flow tracks pixels of similar grayscale values; *v* - video frame, *p* - pixel, defined by location *(x, y)* and time *(t)*.

of a *skilled human observer* that we try to avoid. Avoiding this kind of human bias is the main reason behind the introduction of an automatized data pre-classifying step.

7.1.2. Automated Labelling of Videography

As mentioned in the introduction, the solution to automatic feature tracking are optical flow algorithms. They attempt to interpret the projection of 3D movement onto the 2D imaging plane as a vector field. Variables described by the vector field are either the estimated pixel displacements FIGURE 7.1 or "the velocities of movement of brightness patterns in an image" [73]. Optical flow includes the information contained in the time dimension (Δt). Optical flow algorithms can be categorized into sparse and dense algorithms. Sparse optical flow calculates the vector field for a small set of points of interest (as for example determined by one of those feature detection algorithms). Dense optical flow, on the other hand, calculates the vector field for every single pixel in a video frame by frame [71].

7.1.3. Neural Network Training Using Optical Flow

Initial attempts at using sparse optical flow relied on HOG feature detection. For the detailed source code *see* FUNCTION C. HOG features were used because of their successful history in natural object detection, especially faces [30, 31, 102]. This approach was abandoned after it became clear that the features detected by our HOG computation were mainly whiskers FIGURE 7.2 (a). At this point of the project we aimed at detecting

the movement of complete orofacial behavior, so detecting whiskers only was unsatisfactory. In hindsight, after using HOG features for the final algorithm and learning from DOLENSEK et al the issue at this point was suboptimal parameter tuning of the HOG computation, especially the `BlockSize` FIGURE 7.2. `BlockSize` determines how many pixels are used to calculate the HOG from. DOLENSEK et al used a 32 by 32 pixel square FIGURE 7.2 (b), which covers roughly 1/96 of the our face ROI. At this early point, we used the preset value of `BlockSize` = 8. The resulting 64 pixel square covers a much smaller portion of the mouse face, meaning that the HOG summary is computed on finer features. Fine features in this size range are whiskers and edges, mainly FIGURE 7.2 (a).

Dense optical flow was thus deemed to be the more promising alternative. Dense optical flow estimates the displacement of every single pixel from frame to frame of a video recording. Farnebäck optical flow is the most commonly used dense optical flow algorithm [44]. It relies on a hierarchical abstraction of video features FIGURE 7.3 to speed up computation. Our approach was to track the displacement of pixels of interest over the course of the whole video, extracting their location at any given timepoint. To determine those pixels of interest we calculated mean and variability of the magnitude of the displacement vector displacement vector of every pixel throughout the timecourse of the whole video. Originally we planned on taking the n pixels with the most varying displacement vectors. Quickly it became clear that the pixels determined that way were also located at the whiskers and the edges of the face FIGURE 7.4(b). Again, the initial resolution scale we tried to probe (single pixels) was probably too small to really extract those components that actually make up orofacial movement. In addition, whisking is the most prominent component of orofacial movement in our case since the head-fixation bar impairs movement of the ears, and licking is very sparse, because we did not provide any additional incentive for the animals to perform this behavior (for example no drinking spout). Trying to determine a range of displacement vector variability values from a histogram of all these values that corresponds to observed movement was abandoned, because we did not find a non-arbitrary way of doing so.

7.1.4. Discussion

In conclusion the attempt of using optical flow to extract movement patterns from the mouse face failed because we were not able to extract meaningful features from the video. An important improvement could be to try and apply similar algorithms with a resolution scale better matched to the movement we were trying to capture. Using DLC in combination with optical flow algorithms would most likely be unnecessary, if these algorithms could already be tuned to track features of interest.

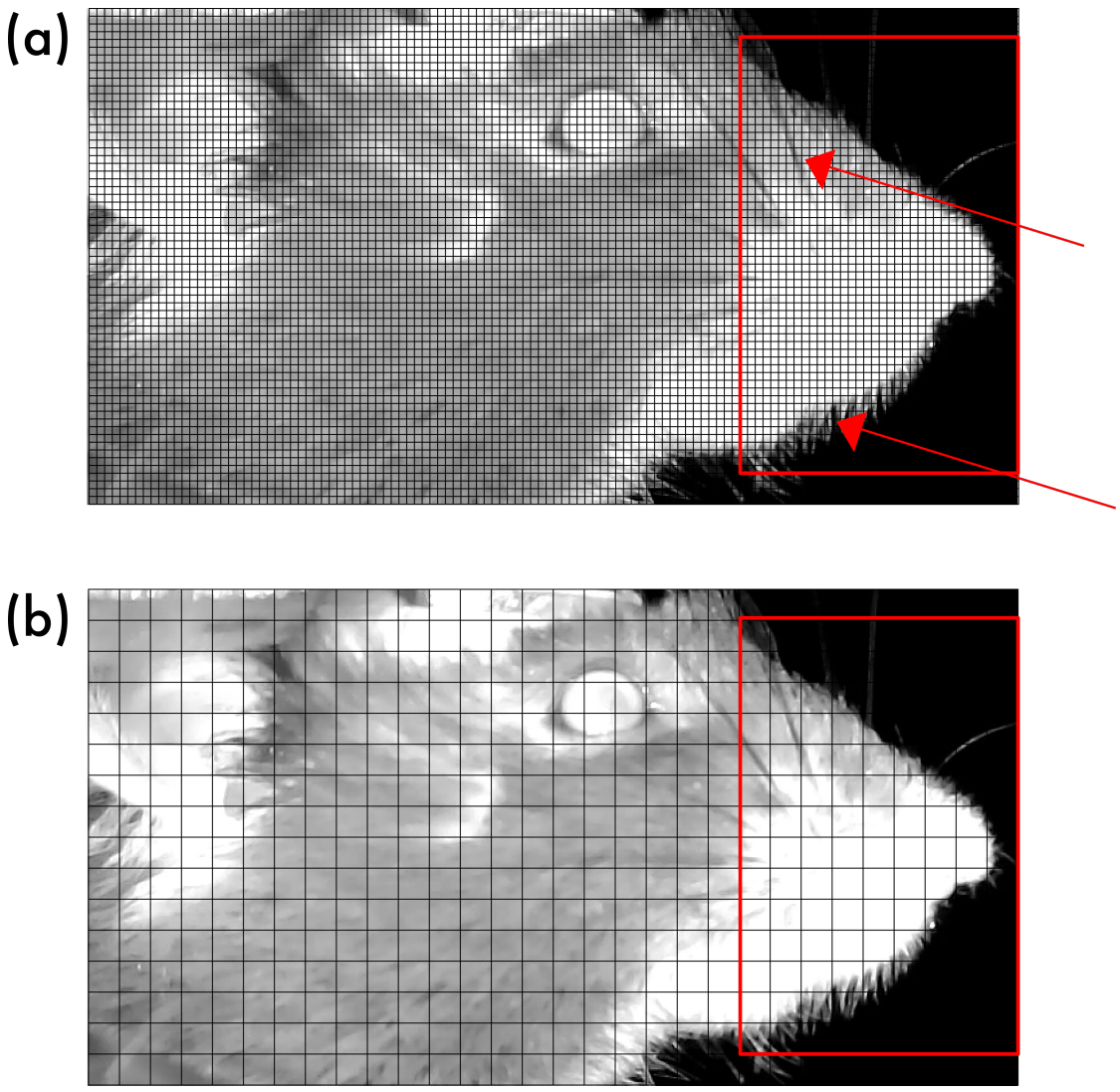

FIGURE 7.2.: Exemplary comparison of two different `BlockSize` values when computing HOGs of a single frame. (a) The initial setting of `BlockSize` = 8 covers only small parts of the ROI (highlighted in red). If any edges fall into a single block they stem from whiskers or edges of the face (red arrow). (b) For `BlockSize` = 32 our ROI is summarized by 96 blocks. Movements of parts of the face, rather than single edges, are better summarized.

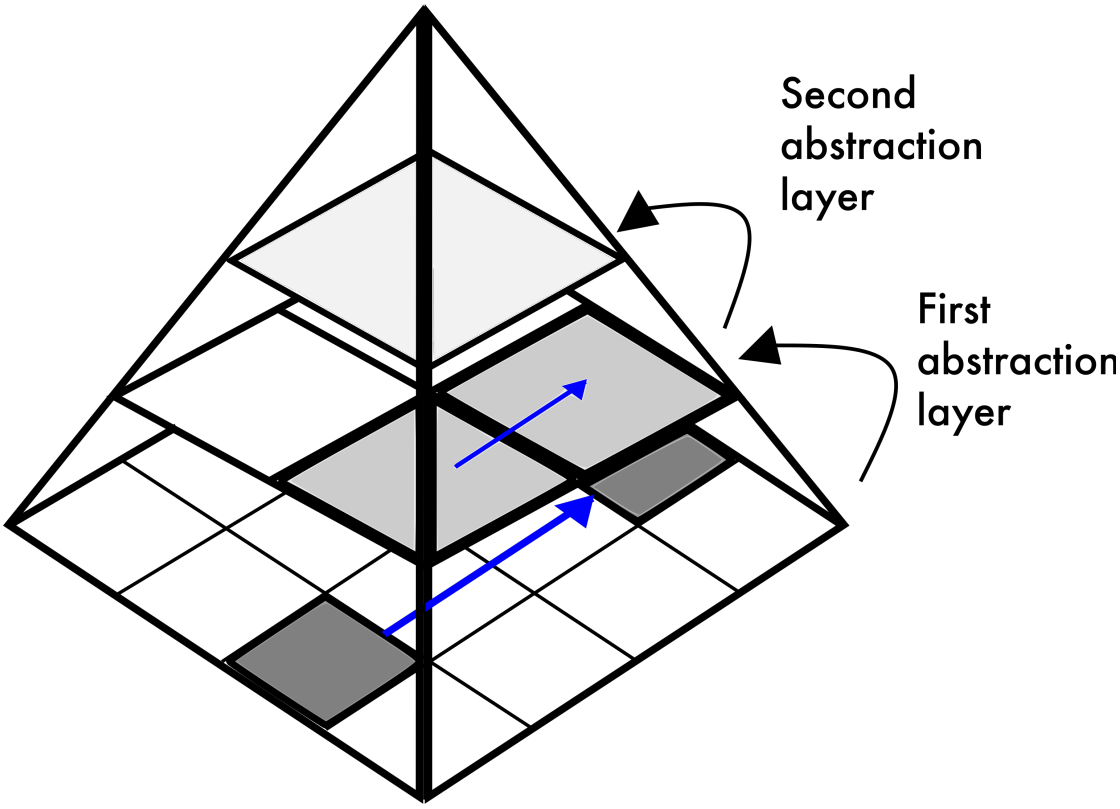

FIGURE 7.3.: Hierarchical abstraction of single pixel displacements in the MATLAB implementation of the Farnebäck optical flow

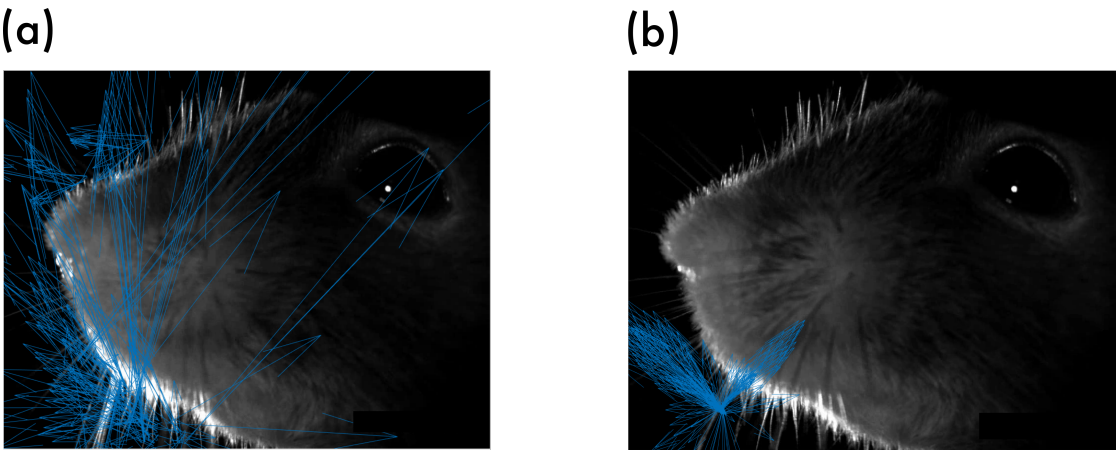

FIGURE 7.4.: Example of dense optical flow applied to determine face regions that move the most. (a) Whiskers are the regions of the highest contrast, because they appear as a clearly delineated black edge in the video. (b) The 200 motion energy vectors with the highest variance of magnitude belong to a single, well illuminated whisker.

7.2. Singular Value Decomposition and Frequency Analysis

Since direct analysis of single pixel values in the video recording was not successful, we directed our attention to a number of dimensionality reduction techniques that might extract more meaningful metrics.

7.2.1. Theoretical Background

Inspired by the tools developed by STRINGER et al (FaceMap) [101] and BERMAN et al (MotionMapper) [9], we were wondering whether it would be possible to fuse those two established libraries together. FaceMap is an algorithm developed to extract different metrics of mouse face video recordings, using singular value decomposition (SVD). It first computes the *Motion Energy* of the video, meaning the absolute frame-to-frame grayscale value difference is calculated for each pixel. SVD is then applied to the 3D tensor holding the motion energy values, extracting the first 1,000 principal components of that tensor. STRINGER et al used reduced rank regression of those traces to explain the variance of recorded neural activity. Reduced rank regression is a linear regression approach where the rank of the regression weights matrix is restricted. By systematically restricting the rank to a range of values, one can find the optimal rank of the regression weight matrix to find the regression weight matrix that minimizes the loss function of the regression. This weight matrix, is then used to predict the neural activity. They determined a rank-16 matrix to be optimal in minimizing the regression error, meaning that 16 parameters (that is the first 16 PCs) well explain the observed variance in the neural activity. 16 PC traces explained 21% of the variance of visual cortex activity observed. For our desired application the code lacked an extraction of stereotyped behavior patterns, since our aim is to understand whether stereotypy of spontaneous orofacial movements increases and whether population coding of this behavior in motor cortex evolves over time. MotionMapper aims at extracting that stereotypy, but from two-dimensional behavior metrics derived from video recordings of drosophila. It applies a Morlet Wavelet transformation to those traces, which obtains the power of these traces in different frequency ranges. This high-dimensional intermediate is then mapped into two dimensions using t-SNE [74]. Gaussian kernel smoothing is used to transform this discrete point cloud into a continuous probability density. Stereotyping is done by "climbing up the gradient of probability density always leads to the same local maximum, often referred to as a watershed transform..." ([9]). Essentially, a watershed transform delineates all local maxima in the probability density from each other. Information of the time dimension is not lost at this point, so that a location

on the two-dimensional probability density can be determined for any timepoint. All frames of the original video recording that fall into the same area delineated by the watershed transform are assigned to the same cluster. Each cluster in turn corresponds to one stereotyped behavior. They were able to prove the validity of this assignment by visual inspection.

7.2.2. Application to Mouse Face Recordings

Performing SVD on a grayscale video recording would, in principle, not require the use of FaceMap. However, FaceMap provides the user with a graphical user interface and is proven to work well. Opposite to the original application, development of the algorithm could not rely on neural activity recordings, because they were not made yet. The first issue with our data was that the scree plot of the SVD did not show any obvious cutoff point for the number of PCs to use with the next analysis steps. It also indicates that the underlying variability is not well explained by the SVD. FIGURE 7.5. Since FaceMap only computes the first 1,000 PCs we decided to use half of them, explaining more than 93% of the variance observed. MotionMapper was able to operate on this output array after some parameter tuning. In our first preliminary analysis, we could not obtain a classification in stereotyped behaviors that matched the observed behavior.

7.2.3. Discussion

In principle this approach should be applicable to our data. MotionMapper is a library that can be applied to any kind of data, as BERMAN et al note. The downside with it is the large number of tunable parameters and the deep nesting of functions. While they avoid binding in too many different MATLAB libraries by this, extensive analysis of the underlying implementation of the math is made more difficult. On top of that the computation done by MotionMapper is very memory-intensive. It is possible that a more thorough revision of the code, including partial re-implementation to further tune it to our aim could eventually be successful. Alternatively, post-hoc verification of the results by comparison with our current findings using a random forest classifier could be tried in the future. If the cluster assignments and their resulting activity pattern vectors match roughly, it would further strengthen the validity of both approaches. As our neural activity recordings are performed by now, it could be analyzed whether our videography PCs explain the variance of the neural activity better than the variance of the video itself.

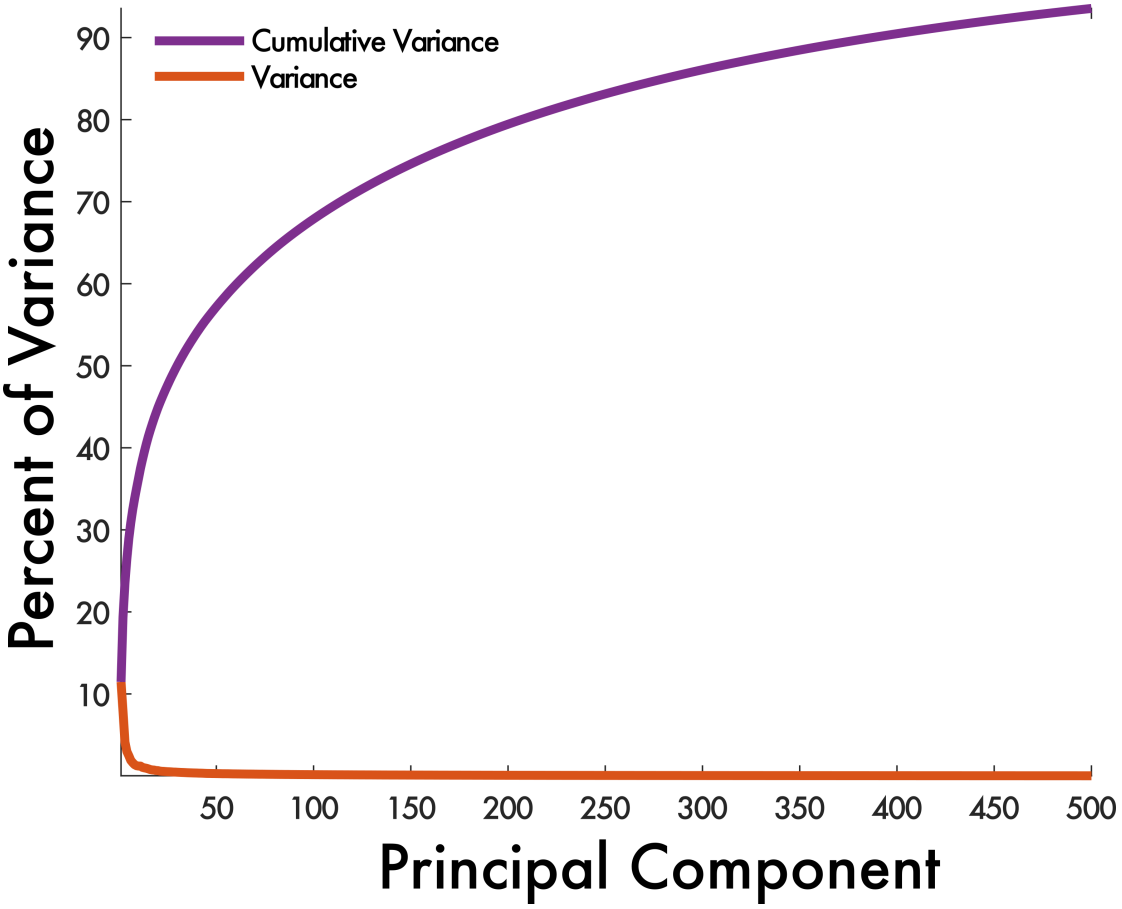

FIGURE 7.5.: Scree plot displaying the first 500 PCs from an exemplary video recording. Apart from the step in variance from the first to the second PC, no obvious shoulder is visible.

Part III.

Discussion

8. Discussion of Current Approach

8.1. Summary

In the course of this thesis an automated grayscale video analysis tool, *FaceCat* that is able to classify the video frames and extract stereotyped patterns has been established. The tool is already being used for the analysis of the experiment it was designed for, were we aim at determining the role and plasticity of M1 in spontaneous face motor activity in head-fixed mice. It is currently published in a GitLab repository. Here I will describe in greater detail the immediate applications, possible improvements and wider applications of the library.

8.2. Immediate Applications

The current aim for which this library is written is to study encoding and plasticity of motor cortex in controlling facial movements. For this head-fixated mice are already recorded with simultaneous video and two-photon Ca^{2+}-imaging. The results of the correlation of those two datasets will be finished in the very near future. Generally, for any future experiments involving head-fixed mice, the addition of a small webcam to the setup can easily be done and would introduce a new interesting dimension to the analysis. A general advantage, when comparing FaceCat to NNs, is the lack of a training bias. As long as all video recordings of an experiment are done from a similar camera angle, FaceCat will output an objective classification of each video frame. The constraint concerning the camera angle is important, because HOGs depend heavily on it (they are not affine-invariant). From the viewpoint of animal welfare, a system that assisted in automated tracking of the animal's wellbeing could be easily developed from the current library. This would be especially useful in experimental contexts where the animal can not be directly observed by the experimenter and needed to be monitored using a camera anyway. For real-time output of the facial activity, a classifier trained on the library's output would be the best choice.

8.3. Next Steps

Immediate next steps for the final library that we decided on using are kept up to date on the library's GitLab repository. Currently open issues are to find a less arbitrary way in how to determine the final number of clusters to assume in the final step of the classification and to include a condition that check whether or not the recording contains IR laser imaging. At the moment the corresponding function needs the parts of the video to display a illuminated eye to determine to determine that they are significant, because two-photon Ca^{2+}-imaging was happening simultaneously FIGURE 6.3 (a) FUNCTION A.2. After these minor amendments, there is the bigger issue that classifications in between different recordings and even recording epochs are not comparable. To solve this issue a classifier of any kind can be trained on the data of one video that assigns the same classifications to any other video to which it is applied. The simple solution is training a random forest classifier like DOLENSEK et al did, alternatively during this thesis I also wrote a convolutional neural network in python's pyTorch library that should be able to perform the same task FUNCTION B. Last, the current implementation is written in MATLAB and dependent on some of its libraries. A re-implementation or port to a true open source solution would offer a larger number of people (especially students at universities not located in Western Europe or the USA) access to the library. python code could also be more prevalent, if the current trend of the field to turn towards python continues.

8.4. Improvements for Wider Application

An important shift in the recording of behavioral experiments at the moment is 3D videography [39, 121]. It is possible to infer 3D structural information by applying optical flow analysis on 2D videography data obtained with two cameras [107] by triangulation. Clearly, considering the real-world third dimension in the analysis is an improvement, since the brain encodes motor behavior in three dimensions[94]. 3D video reconstruction will help to to identify the range and nature of spontaneous face movements in more detail, which would increase the quality of the information of approaches like the one presented in this thesis. For example, our analysis records the mouse face from a viewpoint orthogonal to one side of the face, since movements of the nose and whiskers are recorded with the best contrast. Reconstruction of video data into three dimensions could allow for recording from a more frontal angle, maybe even recording the whole face at once. Less superficial movements, such as movements of

parts other than the whisker pad or swallowing might also be recorded. Deeplearning tools for markerless pose estimation were developed to identify spontaneous behaviors in freely moving animals. Working with head-fixed mice is a necessity for the multi-photon Ca^{2+}-imaging, but also makes video recording and -analysis much easier. Lifting that restriction would allow monitoring animal welfare of caged animals, for example when being singly housed after surgery. A very interesting direction would be the investigation of social meaning for face movements in rodents. This improvement would be a big step from how the algorithm is implemented right now. HOG features are not affine-invariant which means that they are dependent on the viewpoint of the recording camera. When imaging freely moving animals either by a single camera that covers the whole cage area, by a single camera that is able to follow movements by repositioning itself or by having multiple cameras covering the ROI, another method of reducing the dimensionality of the information obtained would need to be implemented. Hessian and Harris operators specifically can be improved with this aim in mind [34].

8.5. The Use of Automated Face Movement Classification in Animal Research

With the advent of more powerful computers and analysis methods the notion to letting the data speak for itself has become more prominent. Rather than collecting only data that matches an a priori hypothesis or tweaking this data accidentally, automated computational analysis can be the approach to find the important results in the unbiased output data itself; BERMAN et al describe this approach:

> Furthermore, these analyses assume, *a priori*, that stereotyped classes of behaviour exist without first showing, from the data, that an organism's actions can be meaningfully categorized in a discrete manner. Ideally, a behavioural description should manifest itself directly from the data, based upon clearly stated assumptions, each with testable consequences. ([9]).

In animal research, especially those subfields concerned with affect, emotion, but also motor behaviors that bear special significance for humans, it is easy to fall victim to anthropomorphic assumptions. In the introduction I pointed out that even human interpretation of other human faces often is flawed, since the causal link between a facial expression and the internal state of an individual has not yet been definitely proven. The transfer of this assumption on other species is yet more problematic, even though it appeals to the assumption originally formally introduced by DARWIN. Today, the

potential to first look at large collections of data, extract patterns from it and only at that point check the implications for a given hypothesis, provides the opportunity to decrease the influence of such assumptions on the outcome and interpretation of an experiment to a minimum. On the most basic level, these objective findings will be able to improve animal welfare not only in the more obvious cases such as pain research, but possibly also in situation such as social isolation, social stress by conspecifics and the experiment's environment (lighting, influence of experimenter, change of location and such). Eventually, finer knowledge of the animal's state and possibly also the reasons for it, will increase the informative value of animal behavioral experiments.

9. Ethical Considerations on the Use of Facial Movement Recognition

Analysis of motor behavior of other individuals is a task only performed by humans themselves up to recently. Implementing above human level processing power to perform the analysis has the ability to be one of the technological developments with great power to transform our society. Classification of other individuals based on appearance, physical characteristics (especially of the face and skull) or certain patterns of motor behavior, however, has a problematic history regarding both ethics and efficacy. Examples of pseudoscientific practices, often used to justify racism, include Phrenology, Physiognomy, Craniotomy and similar practices.

This section first considers some of the discussion points about emotions, their possible expression through motor behavior and their universality, before mentioning current applications that often are lacking in scientific foundation or -consideration. Finally I explore some possibilities of the technology and some thoughts on how it could be implemented in a responsible manner. A current recommendable brief review on this topic was published in [59].

9.1. Western Emotion Categories are Most Likely Not Universal

In the last years, strong doubt has been cast on the basic emotion theory. The six emotions hypothesized to be globally universal (Happiness, Fear, Sadness, Anger, Surprise and Disgust), originally defined by Silvan Tomkins, Paul Ekman and Carroll Izard, had been proven in a large number of experiments starting in the 1970s. Only two decades later researchers began to question the methodology of these studies [93, 95]. The two theories established by these psychologists are the *Theory of Minimal Universality* and the *Theory of Constructed Emotions*. Minimal Universality argues that there is no set of emotion prototypes that are universally felt and expressed, but that the identification of the *valence* of an affect is possible across cultures. Feldman Barrett claims that

mental concepts of emotions, shared by people educated in a similar cultural context (for example Western culture) allow individuals to construct the expected behavior of that emotion, so that perceived motor behavior is matched to the emotion concept applied to a social situation. This is essentially the opposite of recognizing another individuals facial expression as the expression of a certain emotion. Since RUSSELL pointed out the underlying experimental flaw of many basic emotion experiments is priming the experimental subjects to the set of six Western emotion words or only allowing subjects to respond wihtin these terms[93], a number of intercultural studies has found no evidence for the universality of these six emotions. Many cultures do not even have matching translations of these English emotion words, prohibiting the same classification of facial conformations. Studies in cultures that do have matching translations, while not being exposed to Western culture and media, showed that those emotion labels/terms are not applied in the same manner. In a study on the emotion concepts of the Namibian Himba ethnicity, whose language, a dialect of Otji-Herero, contains matching translation of the six basic emotion terms GENDRON et al point out that

> U.S. participants more generally named their piles [of classified photos of facial expressions] with a wide array of additional mental state words such as *surprise* or *concern* ... In contrast, Himba participants were more likely to label face piles with descriptions of physical actions such as *laughing* or *looking at something*, suggesting that individuals in Himba culture used action descriptors more frequently than mental state descriptors to convey the meaning of facial actions.

This hints at the possibility that different cultures not only use different words to describe emotions, but also interpret face movements in a different manner. The idea that face movements are the articulation of an internal emotion might be limited to Western culture and societies influenced by it. There are even historical accounts of the fact that face movements nowadays considered part of the canon of facial expressions, like smiling to convey happiness to others, have been acquired at some point, rather than having been part of human communication ever since [106]. The notion that face movements are not a reliable singular indicator of an individual's internal emotional state has been written down as early as 1542. LEONARDO DA VINCI, giving advice on how to depict emotions in drawings in his work *Tratto della pittura*, wrote

> *Of laughter and weeping and the difference between them.* You will not give the face of one who weeps the same expression as one who laughs although (*in*

> *reality*) they often resemble one another; for it is best to differentiate them [in your drawings], just as the emotion of laughing is different from the emotion of weeping. ([23]).

This section is acknowledging that it can be impossible to infer internal emotions from face movement alone is mentioned together with the centuries-old wish to be able to do so. Up to today the desire that emotions as diverse as happiness and sadness surely must have externally distinguishable expressions influences scientists and engineers alike.

9.2. Facial Expression Recognition Implemented With Little Scientific Foundation

Virtually all of the market leaders in emotion recognition AI software (Affectiva[78, 79], Eyeris[43], Visage Technologies[103], Nemesysco[86], Microsoft[33]) employ models trained on the assumptions made by FACS, adopting the basic emotion theory. A notable exception is Sensum[11], that argues that information from one source (audio, face video or other sensors) is not enough to infer the internal state of an individual[10]. Nemesysco has to be mentioned as an example of particularly bad scientific practice; their listed scientific resources are published in dubious, untranslated journals such as the *Armenian Journal of Mental Health*. Some of the publications seem to be listed twice, from multiple sources. Apart from this bad example, diversity in the target and test groups is often an aim during development and use of the applications. The intrinsic shortfalls of the basic emotion theory regarding diverse cultural backgrounds (and their possibly different ways to communicate internal states), at the same time, is rarely discussed in detail. Unfortunately, many of these companies publish their studies on a private basis, which means these publications essentially lack scrutiny from the wider scientific community. Application of these technologies range from maximizing the effectiveness of ads, of electoral campaigns, to assisting interactions with AI, psychological diagnosis and even law enforcement. Arguably a lot of this is done based on outdated scientific findings from the 70s. This divergence between actuality of the computational methods on the one hand and the psychological and neuroscientific foundations on the other has to be questioned. Considering the potential of algorithms in this field in transforming the way we perceive and communicate with each other, it is to be hoped that Chief Technology Officers and their development departments will increasingly work together with scientists not only in Psychology and Neuroscience, but also Ethics and Social Sciences.

9.3. The Link Between Perception of Emotions and Facial Motor Actions Needs to be Proven

Emotions might not necessarily be universal and face expressions maybe acquired culturally, but they are part of non-verbal communication at least in Western cultures. The link of face movement and emotion needs to be investigated carefully. There might be a set of basic emotions on which members of Western culture agree, but assuming that fact confounds studying possibly related expressions. Ideally, the link could be established from the opposite direction: What is the neural activity causing the observed face motor behavior? Here, one has to be careful not to define an emotion by a certain motor behavior. Behavioral Neuroscience often avoids defining the term *emotion* itself by describing emotion-related behavior instead of the emotion or the underlying brain state itself. Face movement recognition algorithms can be used to establish causality between face motor behavior and the experimental subject's emotion state. Analysis of brain activity in multiple experimental contexts can be used to determine a correlative mapping between the stimulus and the activity. After this mapping of context/stimulus to brain state a second mapping from face motor behavior to brain state can be established. This correlation of context to brain state to face movement could then be probed in the opposite direction. In the end, the context needs to be labelled with an associated emotion. In studies involving human subjects interviews revealing the subjects internal state can be used. In the case of animal experiments a human emotion category needs to be used as a label. The more diverse the investigated contexts the better this mapping of human emotion category to the actual brain activity of the animal. Most likely, animals can not experience emotions exactly like humans do, because they are missing the ability to classify their interoceptive status in terms of human language. An approach like this could reveal if there are brain states in animals that closely match human emotions. If no match between a set of contexts that reliably cause a certain spectrum of emotions in humans and animal brain states can be made, This could either show that no emotion concepts beyond simple affect are present or that the animal has an entirely different set of interoceptive brain states/emotions.

The work presented in this thesis certainly is not yet sophisticated enough to pose any of these questions directly, but research in literature and application of similarly aimed algorithms and approaches made it clear to me that often ethical or even scientific questions are ignored in the intense pursuit of solving the engineering side of the problem.

Appendix

A. FaceCat Algorithm

This appendix describes in detail the working its single steps. It is intended both as a reference and a documentation. The steps outlined in ALGORITHM 1 are invoked by the utility function `processWrapper`, into which the user needs to enter the paths to the video files to be analyzed.

Algorithm 1 FaceCat Algorithm

▷ Converting the video frames into HOG vectors

load video
select ROI of snout region
select ROI of eye
$frame = recording\_idx \leftarrow$ detect recording epoch based on grayscale value mean of eye-ROI
for $frame = recording\_idx_0$ to $recording\_idx_{end}$ **do**
    cut out only snout-ROI
    convert that image to grayscale, if necessary
    compress that image to HOG vector
    store HOG vector as row entry in a $HOGmatrix$
end for
$distanceMatrix \leftarrow$ Calculate the cosine distance between all rows in the $HOGmatrix$

▷ Processing the distance matrix to cluster frames

Remap all HOG vectors into a 2D space using t-SNE
Convert the discrete point cloud into a continuous distribution using a Gaussian kernel $\sigma = 1.8$
Perform an inverse watershed transform on the resulting pdf
$numberOfClusters \leftarrow$ Count the number of delineated regions
Generate the linkage of the $distanceMatrix$
$classificationvector \leftarrow$ cluster the linkage, using $numberOfClusters$
Save the resulting vector containing one assigned cluster number per frame

▷ Extracting stereotypy

Extract all bits of length $windowSize$ from the cluster assignment vector and store them in a matrix
Delete all duplicate entries from that matrix
Delete all cirular shifted duplicates from that matrix
if Option 1: $windowMode = 'discrete'$ **then**
    Downsample the cluster assignment vector by a factor of $windowSize$
else if Option 2: $windowMode = 'continuous'$ **then**
    pass
end if
Convolve all remaining patterns with the resulting cluster assignment vector
for $pattern = 1$ to end **do**
    for $i = 1$ to $recording\_idx_{end} - windowSize$ **do**
        count the number of equal values
        divide the result by $windowSize$
    end for
    Determine the number and timepoints of re-occurrence of that pattern
    Store results in matrix
end for

A.1. Materials and Methods

A.1.1. Programming

All MATLAB code was written using MATLAB versions 9.7 and 9.8. The code relies on the following packages: Image Processing Toolbox 11.1, Statistics and Machine Learning Toolbox 11.7, Computer Vision Toolbox 9.2, Parallel Computing Toolbox 7.2, MATLAB Parallel Server 7.2. The python CNN was written using python version 3.7.1 including the pytorch 1.1.0, torchvision 0.2.2, numpy 1.15.4, pandas 0.23.4, scikit-image 0.14.1, pillow 5.3.0, imageio 2.4.1 packages.

A.1.2. Face videography setup

The mice were singly-housed in their home cage during experiments. During video and Ca^{2+}-imaging recordings, the mice where placed into a body jacket (custom-made plexiglass tube) and their headbar srewed to a headbar-holder. The tube was large enough in diameter for the mice to move their front limbs freely. A consumer-grade IR webcam (webcamera_usb, 1/2.7" CMOS OV2710 sensor, 1080 x 720 px resolution at 60 Hz) was placed on a monopod fixed to the same stage on which the mouse was placed, ensuring an orthogonal recording angle to the side of the face. Recording sessions were limited to a maximum of 20 min. Imaging software used was Windows Camera.

A.2. Pre-processing: Converting the Video Frames Into HOG Vectors

First, the first frame of every video is loaded into a separate window, allowing to specify the ROIs for both the eye and the snout. The pixels within the ROI of the eye are analyzed by `measureGrayValue` FUNCTION A.1, which extracts the maximum grayscale value within that ROI for every frame. This information is passed to `extractRecordedFramesIdcs` FUNCTION A.2, that puts a hard threshold of 200 on this list of values, to determine whether the eye was white during the corresponding frame. Frames displaying a white eye either belong to a recording epoch or are outliers, caused by exploratory IR laser scanning or reflection effects. To distinguish these, the function checks if the eye was illuminated by IR light for at least one minute continuously. `stitchthresholdHoles` FUNCTION A.3 is an additional helper to determine and include sequences of less than 50 frames that show grayscale values below the threshold of 200, to avoid over-segmentation of the recording epochs due to low-end outliers.

The output at this stage are two lists containing the onsets and offsets of two-photon Ca^{2+}-imaging recording epochs.

Each of these recording epochs is analyzed sequentially in the core function `batchExtractHOG_concat` FUNCTION A.4. Each frame is converted into a grayscale image, if necessary, and only the specified snout ROI is kept for HOG computation. With the presets used for our analysis the ROI always had a size of 381 by 251 pixels, resulting in HOG vectors of length 616 per frame. All these vectors are stored as one row in a matrix. Once the iteration through the recording epochs is done, a cosine distance matrix is calculated from these matrix rows. This matrix is the foundation for the classification of the frames, since similar frames will have resulted in similar HOG vector representations that in turn have a short cosine distance. Clustering will be done hierarchically, so a linkage of the cosine distance matrix is generated at this point. HOG vector matrix, distance matrix, and linkage are stored as a result of this function.

```
function grayValue = measureGrayValue(video_path, pos_eye)
  vidReader = VideoReader(video_path);
  grayValue = [];
  while hasFrame(vidReader)
    current_frame = readFrame(vidReader);
    eyeArea = grayCrop(current_frame, pos_eye);
    tmp = regionprops(ones(size(eyeArea), 'logical'), eyeArea, ...
                      'MaxIntensity');
    tmp = struct2cell(tmp);
    % find empty cells
    emptyCells = cellfun(@isempty,tmp);
    % remove empty cells
    tmp(emptyCells) = [];
    % calculate the grayscale value mean
    tmp = mean(cellfun(@(x) x(:), tmp));
    % append measurement to vector
    grayValue(end+1,1) = tmp;
  end
end
```

FUNCTION A.1.: `measureGrayValue` calculates grayscale average of maximum intensity pixels.

```
function stereotypedFrames = extractStereotyped(cossim_hogs, cutoff)
  %% Assign a cluster number to every stillframe of the video
```

```
3             links = linkage(cossim_hogs, 'average');
4             stereotypedFrames = cluster(links, 'Cutoff', cutoff, 'Criterion', ...
5                                         'distance');
6         end
```

FUNCTION A.2.: Calculate `classification vector` (`stereotypedFrames`) by clustering the distance matrix using the cutoff value.

```
1         function laserOn = stitchThresholdHoles(laserOn, laserSwitch)
2             % This bit makes sure that single frames that fall below the threshold
3             % don't impact the separation of recording epochs
4
5             % Where are laser switch on events detected?
6             changes = find(laserSwitch == 1);
7             % How long are the corresponding record-off snippets?
8             snippetLengths = diff(changes);
9             % All snippets shorter than 50 frames are probably just outliers,
10            % not true switch-off events
11            shortBits = find(snippetLengths < 50);
12            shortBitStart = changes(shortBits);
13            shortBitEnd = shortBitStart + snippetLengths(shortBits) - 1;
14            if ~isempty(shortBitStart)
15                for bitIter = 1:length(shortBitStart)
16                % replace the outlier holes with the value just before they start
17                laserOn(shortBitStart(bitIter):shortBitEnd(bitIter)) = ...
18                  laserOn(shortBitStart(bitIter) - 1);
19                end
20            end
21        end
```

FUNCTION A.3.: `stitchThresholdHoles` checks if the grayscale value of short sequences of frames fall below the threshold value of 200.

```
1 function batchExtractHOG_concat(video_path, laserSwitchOn_idcs, laserSwitchOff_idcs, ...
2             batchNum, pos_snout)
3     vidReader = VideoReader(video_path);
4     % Extract File Name, for savename generation
5     try
6         filename = regexp(video_path, '[/\\](\w+)\.', 'tokens');
7         filename = filename{1}{1};
```

```
8       catch
9           disp('Remove any special characters from the file name');
10          keyboard
11      end
12      dataChunks = length(laserSwitchOn_idcs);
13      % Concatenate the indices of all frames during which the Ca recording
14      % laser is on
15      recordedFrames = [];
16      for N=1:dataChunks
17          tmp = laserSwitchOn_idcs(N):laserSwitchOff_idcs(N);
18          recordedFrames = [recordedFrames tmp];
19      end
20      disp(strcat('Processing chunk_', num2str(N), ' out of_ ', num2str(dataChunks)));
21      hog_ChunkN = single.empty;
22      for frame=1:length(recordedFrames)
23          % read the frames and convert them into HOG vectors
24              img = read(vidReader, recordedFrames(frame));
25              img = grayCrop(img, pos_snout);
26              hog_vec = extractHOGFeatures(img, 'CellSize', [32 32], 'NumBins', 8, ...
27                                  'BlockSize', [1 1]);
28              hog_ChunkN(end+1, :) = hog_vec;
29      end
30      try
31          cossim_hogs = pdist(hog_ChunkN, 'cosine');
32          links = linkage(cossim_hogs, 'average');
33      catch
34          disp('No distance calculation');
35          save(strcat('Output_', filename, '.mat'), '-v7.3', 'hog_ChunkN');
36      end
37          disp(strcat('Saving results for chunk ', num2str(N)));
38          save(strcat('Output_', filename, '.mat'), '-v7.3', 'hog_ChunkN', ...
39                      'cossim_hogs', 'links');
40          disp('saved');
41
42
43  end
```

Function A.4.: `batchExtractHOGconcat` is the main function, that loads frames, cuts out the selected ROI and calculates a HOG for that frame and ROI.

A.3. Processing the Distance Matrix to Cluster Frames

`completePostAnalysis` FUNCTION A.5 is the second wrapper function, performing the actual clustering steps. The user needs to specify the paths to the output generated earlier. To obtain an estimate of the number of clusters present in the distance matrix, an alternative clustering approach is being used on the HOG vector matrix. First, every HOG vector is embedded into a two-dimensional space using t-SNE (all default parameters of the MATLAB implementation of t-SNE are kept here.). Then a continuous pdf is generated based on the point cloud by applying a Gaussian smoothing kernel of width `sigma` in every dimension. `make_density_plots` and `findPointDensity` perform this step and are direct copies from the MotionMapper library. This smoothed distribution can then be clustered by applying a watershed transform to the density plot and counting the number of distinct areas delineated this way `inv_watershed`. As mentioned before, these steps are our implementation of the clustering performed by the MotionMapper library [9]. The number of clusters estimated this way is then used to find a threshold for the linkage generated earlier. The resulting output `classifiedFrames` is a vector of length equal to the number of frames of the recording epoch in question. Every entry is a classification number of that frame.

```
function completePostAnalysis(sigma)
    % list of all locations of hog_ChunkN
    paths = {    ...
    '/path/to/processWrapper/output.mat' ...
    };
    if isempty(paths)
    disp('Create a cell array "paths" cotaining the full paths of the data ...
         to be analyzed');
         keyboard
    end

    % Perform all post analysis steps and store the results in the same
    % directory as the data
    for i = 1:length(paths)
        path = paths{i};
        try
            load(path, 'hog_ChunkN', 'links')
        catch
            disp('Adjust file name, path not found');
            keyboard
```

```
21          end
22          % Extract experiment ID
23          [filepath,name,ext] = fileparts(path);
24          % generate tSNE mapped 2D representation of the HOGs
25          tSNEmap = tsne(hog_ChunkN);
26          [~, ~, boundaries, watershed_map] = watershedClustering(tSNEmap, ...
27                                                      sigma);
28          % Length of boundaries is the number of clusters detected
29          noClusters = length(boundaries);
30          classifiedFrames = cluster(links, 'maxclust', noClusters);
31
32          % Generate output for both options, consecutive and sliding window
33          % analysis of the cluster patterns
34          windowSizes =  [15 30 45 60];
35          for j=1:length(windowSizes)
36              windowSize = windowSizes(j);
37              [patComp] = detectPatterns(classifiedFrames, 'windowSize', windowSize, ...
38                          'minOverlap', 'boot', 'windowMode', 'distinct');
39
40              save(strcat(filepath, '/', name, num2str(noClusters), '_window_', ...
41                  num2str(windowSize), '_distinct_.mat'), 'patComp', 'watershed_map', ...
42                  'boundaries', '-v7.3')
43
44              [patComp] = detectPatterns(classifiedFrames, 'windowSize', windowSize, ...
45                          'minOverlap', 'boot', 'windowMode', 'sliding');
46
47              save(strcat(filepath, '/', name, num2str(noClusters), '_window_', ...
48                  num2str(windowSize), '_sliding_.mat'), 'patComp', '-v7.3')
49          end
50      end
51  end
```

FUNCTION A.5.: `completePostAnalysis` first performs classification of the frames, before detection of stereotyped patterns in the `classification vector` by calling `detectectPatterns` FUNCTION A.6.

A.4. Post-processing the Classified Frames: Extracting Stereotypy

Extracting stereotypy in the sequence of classified frames is done by `detectPatterns` FUNCTION A.6. This function takes a number of arguments from the user, specifying the size of the analysis window (`windowSize`), the threshold when to consider two sequences *equal* (`minOverlap`) and the convolution method (`windowMode`), either *distinct*, resulting in a distinct sequential convolution of each unique pattern with the classification vector or *sliding*, calculating the overlap between pattern and classification vector at every entry of the classification vector *see also* FIGURE 6.4. By default, the threshold for equality is calculated using randomized resampling of the patterns 10,000 times, calculating the overlap of those patterns every time (also known as *bootstrapping*). From the resulting distribution, the 95% quantile is used as the threshold value. Defaults for the `windowSize` and `windowMode` are 15 and 'distinct', respectively. This is caused by the fact that our video recordings were done with a frame rate of 60 Hz, while the two-photon Ca^{2+}-imaging reached a volume scanning rate of 4 Hz. Analyzing only every 15th value of the classification vector results in extraction of patterns that last 1/60 * 15 seconds, which downsamples the information from 60 to 4 Hz. Patterns are extracted by storing each sequence of numbers of the classification vector of length equal to `windowSize` as a row in a separate matrix. To speed up computation and yield more concise results duplicate patterns are removed from the matrix first. Then, circularly shifted versions of the same patterns are removed. This is done by extending each pattern to contain itself twice in sequence. Iteratively a string comparison finds if each pattern can be found in the extended version of each other pattern. After the removal of any duplicates defined this way, the overlap convolution is performed. Every remaining pattern is iteratively compared to the classification vector entires, either using the 'distinct' or 'sliding' method, as specified above. The overlap metric simply counts the number of equal elements between the pattern and the classification vector segment at this iteration step. The result is divided by the value of `windowSize`, which results in a relative score between 0 (no overlap at all) and 1 (total overlap). Using the threshold for equality, either specified manually by the user or computed using the bootstrap, indices and number of repetitions of the pattern being probed are stored in a separate matrix. The final output is stored as a MATLAB structure array, containing all unique patterns, their respective overlap scores and -locations, the threshold value used to determine equality (or total overlap), the classification vector and the distance matrix TABLE 6.1.

```
function [patComp] = detectPatterns(classifiedFrames, varargin)
%% Find all sequential patterns of length windowSize in classifiedFrames,
%% store their count and index locations
% Set default input values and parse input arguments
    defaultWindowSize = 15;
    defaultWindowMode = 'distinct';
    expectedWindowMode = {'distinct', 'sliding'};
    defaultMinOverlap = 'boot';


    parser = inputParser;
    validVector = @(x) isnumeric(x) && size(x, 2) == 1;
    validScalarPosNum = @(x) isnumeric(x) && isscalar(x) && (x > 0);
    validMinOverlap = @(x) isnumeric(x) && isscalar(x) || strcmp(x, 'boot');
    addRequired(parser, 'classifiedFrames', validVector);
    addParameter(parser, 'windowSize', defaultWindowSize, ...
                 validScalarPosNum);
    addParameter(parser, 'minOverlap', defaultMinOverlap, ...
                 validMinOverlap);
    addParameter(parser, 'windowMode', defaultWindowMode, ...
                 @(x) any(validatestring(x,expectedWindowMode)));
    parse(parser, classifiedFrames, varargin{:});

    classifiedFrames = parser.Results.classifiedFrames;
    minOverlap = parser.Results.minOverlap;
    windowSize = parser.Results.windowSize;
    windowMode = parser.Results.windowMode;
%% Find repeating sequences, output their number and location
    if isnumeric(minOverlap) && minOverlap > windowSize
        error('minOverlap cannot be bigger than windowSize');
    end
    % Extract all possible patterns given the windowMode constraint
    if strcmp(windowMode, 'distinct')
        allPatterns = im2col(classifiedFrames, [windowSize 1], 'distinct')';
    else
        allPatterns = im2col(classifiedFrames, [windowSize 1], 'sliding')';
    end

    % Remove duplicate patterns to improve speed of the convolution
    allPatterns = unique(allPatterns, 'rows', 'stable');
```

```
41      % Remove circular shifts/permutations of patterns
42      allPatterns = removeDuplicatePatterns(allPatterns);
43      % Optionally: calculate cutoff based on bootstrap rather than
44      % specifying the minimum overlap
45      if strcmp(minOverlap, 'boot')
46          cutoff = bootDist(allPatterns);
47          cutoff(3, :) = cutoff(2, :)/windowSize;
48          minOverlap = cutoff(2, 1);
49      end
50
51      % Convert classifiedFrames to a column vector
52      if size(classifiedFrames, 2) == 1
53          classifiedFrames = classifiedFrames';
54      end
55      % Expand classified frames, do all overlap comparisons with the
56      % remaining patterns at once for every frame
57      noPatterns = size(allPatterns, 1);
58      noConvItr = size(classifiedFrames, 2) - windowSize + 1;
59      classifiedFrames = repmat(classifiedFrames, noPatterns, 1);
60      scoreMat = zeros(noPatterns, noConvItr);
61      for i=1:noConvItr
62          scoreMat(:,i) = overlapCount(classifiedFrames(:, i:i + (windowSize-1)), ...
63                          allPatterns);
64      end
65      % Downsample ('distinct' == true) to align with Ca recording
66      % Normalize to obtain score between 0 and 1
67      if strcmp(windowMode, 'distinct')
68          scoreMatConv = scoreMat(:, 1:windowSize:end)/windowSize;
69          classifiedFramesConv = classifiedFrames(1, 1:windowSize:end);
70      else
71          scoreMatConv = scoreMat/windowSize;
72          classifiedFramesConv = classifiedFrames;
73      end
74      % Initialize structure to store the results
75      patComp = struct('Pattern', num2cell(allPatterns, 2), ...
76          'Score', num2cell(scoreMatConv, 2));
77      simMat = patMat(allPatterns, windowSize);
78      patComp(1).Assigned_Clusters = classifiedFramesConv(1, :);
79      patComp(1).simMat = simMat;
80      patComp(1).cutoff = array2table(cutoff', ...
81          'VariableNames', {'Percent Cutoff', 'Corresponding Overlap in Frames', ...
```

```
82          'Corresponding similarity'});
83      % Store the locations for the varying degrees of overlap
84      scoreMat = scoreMat(:, 1:windowSize:end);
85      parfor k = 1:noPatterns
86          itr_count = 0;
87          patComp(k).Overlap_Locations = struct(('Overlaps'), {});
88          for j=minOverlap:windowSize
89              itr_count = itr_count + 1;
90              tmpOverlapLoc = find(scoreMat(k,:) == j);
91              patComp(k).Overlap_Locations(itr_count).Overlaps = ...
92                  struct(strcat('MinOverlap', num2str(j)), tmpOverlapLoc);
93          end
94      end
95  end
```

FUNCTION A.6.: `detectPatterns` is the core function detecting stereotypy in the classification vector.

B. pyTorch Convolutional Neural Network

Below is listed the source code for a CNN able to classify the video data directly, when provided with enough training data. The code was not applied to real-life data yet but runs well on test data. This CNN classifier could be used as an alternative to a simple random forest classifier or to verify other approaches. The advantage of using a separate classifier after having classified the data already is that the output labels, generated by a trained classifier, can be used interchangeably across multiple videos. Ideally, the classifier should be trained on as diverse data as possible (different video recordings from different experimental animals). The structure of the CNN is fairly simple and can be directly read out from the `Net` class FUNCTION B.1. The code was written with the help of the pyTorch manual and documentation pages and contain direct citations from those [24]. Four convolution layers and one pooling layer reduce the original video information to 128*4*9 = 4,608 dimensions. Three linear transformation layers, or fully connected layers (fc), reduce the final output to the number of classes (often referred to as *labels* in the context of neural networks) desired. A custom dataloader class was written to handle the input format and the frame size of the video FUNCTION B.2. It generates a pyTorch tensor, containing one label per video frame. Tensors are the main data type that can be easily handled by the other pyTorch functions. It is also easier to apply GPU acceleration to pyTorch computations, if all variables are expressed as pyTorch-internal data types. Finally, a training function splits the labelled data into a training- and a test-set, trains the CNN and outputs the accuracy achieved. A wrapper script FUNCTION B.4 defines the variables needed: Paths to the labels (`csv_path`), the video (`filename`), the number of training epochs, the dataloader class and the network class (`net`).

```
1 import torch
2 import torchvision
3 import torchvision.transforms as transforms
4 import matplotlib.pyplot as plt
5 import numpy as np
6 import torch.nn as nn
```

```
7  import torch.nn.functional as F
8
9
10 class Net(nn.Module):
11     def __init__(self, no_clusters):
12         super(Net, self).__init__()
13         self.conv1 = nn.Conv2d(1, 6, 5)
14         self.pool = nn.MaxPool2d(2, 2)
15         self.conv2 = nn.Conv2d(6, 16, 5)
16         self.conv3 = nn.Conv2d(16, 64, 5)
17         self.conv4 = nn.Conv2d(64, 128, 5)
18         self.fc1 = nn.Linear(128 * 4 * 9, 1024)
19         # nn.init.kaiming_uniform_(self.fc1, mode='fan_in', nonlinearity='relu')
20         self.fc2 = nn.Linear(1024, 512)
21         # nn.init.kaiming_uniform_(self.fc2, mode='fan_in', nonlinearity='relu')
22         self.fc3 = nn.Linear(512, no_clusters)
23         # nn.init.kaiming_uniform_(self.fc3, mode='fan_in', nonlinearity='relu')
24
25     def forward(self, x):
26         x = self.pool(F.relu(self.conv1(x)))
27         x = self.pool(F.relu(self.conv2(x)))
28         x = self.pool(F.relu(self.conv3(x)))
29         x = self.pool(F.relu(self.conv4(x)))
30         x = x.view(-1, 128 * 4 * 9)
31         x = F.relu(self.fc1(x))
32         x = F.relu(self.fc2(x))
33         x = self.fc3(x)
34         return x
```

Function B.1.: The CNN class used for frame classification.

```
1  import os
2  import torch
3  import imageio
4  import pandas as pd
5  import numpy as np
6  import matplotlib.pyplot as plt
7  from skimage import io, transform ## this does the image resizing
8  from PIL import Image
9  from torch.utils.data import Dataset, DataLoader
10 from torchvision import transforms, utils
```

```
11
12  class MotionEnergyDataset(Dataset):
13
14      def __init__(self, csv_path, avi_path, transform=None):
15          """
16          Args:
17          csv_path (string): Path to the csv file (one column containing
18            cluster assignment for the motE frame with the number
19            of the corresponding row)
20          avi_path (string): Path to the avi file (cropped video of the mouse
21            behavior)
22          transform (callable, optional): Optional transform to be applied
23                  on a sample.
24          """
25          self.labels_frame = pd.read_csv(csv_path)
26          self.vidReader = imageio.get_reader(avi_path, 'ffmpeg')
27          self.frame_size =  self.vidReader.get_meta_data()['size']
28          self.frame_size = (self.frame_size[1], self.frame_size[0])
29          self.transform = transform
30
31      def __len__(self):
32          return len(self.labels_frame)
33
34      def __getitem__(self, idx):
35          if torch.is_tensor(idx):
36              idx = idx.tolist()
37          past_frame = np.zeros(self.frame_size)
38          current_frame = np.zeros(self.frame_size)
39          motE = np.zeros(self.frame_size)
40          if idx == 0:
41              pass
42          else:
43            # Read frame
44            current_frame[:,:] = self.vidReader.get_data(idx)[:,:,0]
45            # Get the frame one prior to current
46            past_frame[:,:] = self.vidReader.get_data(idx - 1)[:,:,0]
47            # Calculate the motion energy
48            motE = current_frame - past_frame
49            # Get the cluster assignment
50          label = self.labels_frame.loc[idx, 'torch']
51          label = np.asarray(label, dtype=np.uint8)
```

```
52          motE = np.asarray(motE, dtype=np.uint8)
53
54          sample = {'frame': motE, 'label': label}
55
56
57          if self.transform:
58              frame, label = sample
59              sample = self.transform(sample)
60
61          return sample
62
63  class _2PILImage(object):
64      """Convert a tensor or an ndarray to PIL Image.
65
66      Converts a torch.*Tensor of shape C x H x W or a numpy ndarray of shape
67      H x W x C to a PIL Image while preserving the value range.
68
69      Args:
70      mode (`PIL.Image mode`_): color space and pixel depth of input data
71      (optional).
72      If ``mode`` is ``None`` (default) there are some assumptions made about
73      the input data:
74          - If the input has 4 channels, the ``mode`` is assumed to be ``RGBA``.
75          - If the input has 3 channels, the ``mode`` is assumed to be ``RGB``.
76          - If the input has 2 channels, the ``mode`` is assumed to be ``LA``.
77          - If the input has 1 channel, the ``mode`` is determined by the data
78          type (i.e ``int``, ``float``, ``short``).
79
80      .. _PIL.Image mode:
81      https://pillow.readthedocs.io/en/latest/handbook/concepts.html#concept-modes
82      """
83      def __init__(self, mode=None):
84          self.mode = mode
85      def __call__(self, pic):
86          """
87          Args:
88              pic (Tensor or numpy.ndarray): Image to be converted to PIL Image.
89
90          Returns:
91              PIL Image: Image converted to PIL Image.
92
```

```
93          """
94          frame = pic['frame']
95          frame = transforms.functional.to_pil_image(frame, self.mode)
96          label = pic['label']
97          sample = {'frame': frame, 'label': label}
98          return sample
99
100
101     def __repr__(self):
102         format_string = self.__class__.__name__ + '('
103         if self.mode is not None:
104             format_string += 'mode={0}'.format(self.mode)
105         format_string += ')'
106         return format_string
107
108 class _2Tensor(object):
109     """Convert a ``PIL Image`` or ``numpy.ndarray`` to tensor.
110
111     Converts a PIL Image or numpy.ndarray (H x W x C) in the range
112     [0, 255] to a torch.FloatTensor of shape (C x H x W) in the range [0.0, 1.0]
113     if the PIL Image belongs to one of the modes
114       (L, LA, P, I, F, RGB, YCbCr, RGBA, CMYK, 1)
115     or if the numpy.ndarray has dtype = np.uint8
116
117     In the other cases, tensors are returned without scaling.
118     """
119
120     def __call__(self, pic):
121         """
122         Args:
123             pic (PIL Image or numpy.ndarray): Image to be converted to tensor.
124
125         Returns:
126             Tensor: Converted image.
127         """
128         frame = pic['frame']
129         frame = transforms.functional.to_tensor(frame)
130         label = pic['label']
131         label = torch.from_numpy(label)
132         sample = {'frame': frame, 'label': label}
133         return sample
```

```
134
135     def __repr__(self):
136         return self.__class__.__name__ + '()'
137
138 class Rescale(object):
139     """Resize the input PIL Image to the given size.
140
141     Args:
142         size (sequence or int): Desired output size. If size is a sequence like
143             (h, w), output size will be matched to this. If size is an int,
144             smaller edge of the image will be matched to this number.
145             i.e, if height > width, then image will be rescaled to
146             (size * height / width, size)
147         interpolation (int, optional): Desired interpolation. Default is
148             ``PIL.Image.BILINEAR``
149     """
150
151     def __init__(self, size, interpolation=Image.BILINEAR):
152         assert isinstance(size, int) or (isinstance(size, Iterable) and
153         len(size) == 2)
154         self.size = size
155         self.interpolation = interpolation
156
157     def __call__(self, img):
158         """
159         Args:
160             img (PIL Image): Image to be scaled.
161
162         Returns:
163             PIL Image: Rescaled image.
164         """
165         frame = img['frame']
166         frame = transforms.functional.resize(frame, self.size, self.interpolation)
167         label = img['label']
168         sample = {'frame': frame, 'label': label}
169         return sample
170
171     def __repr__(self):
172         interpolate_str = _pil_interpolation_to_str[self.interpolation]
173         return self.__class__.__name__ +
174         '(size={0}, interpolation={1})'.format(self.size, interpolate_str)
```

FUNCTION B.2.: The dataloader class provides correct feature (video) and label (`classification vector`, in this case as .csv) input for the CNN and the training function FUNCTION B.3.

```
def train_net(epochs, trainloader, optimizer, net):
    criterion = nn.CrossEntropyLoss()
    cost_list = []
    accuracy_list = []
    for epoch in range(epochs):
        running_loss = 0.0
        for i, data in enumerate(trainloader, 0):
            print(i)
            # 0) Get the inputs; data is a list of [inputs, labels]
            inputs = data['frame']
            labels = data['label']
            labels = labels.type(torch.long)
            # 1) Reset gradients
            optimizer.zero_grad()
            # 2) Make the prediction
            prediction = net(inputs)
            # 3) Calculate loss
            loss = criterion(prediction, labels)
            # 4) Calculate gradient of parameters
            loss.backward()
            # 5) Update parameters
            optimizer.step()
            # 6) Track loss
            running_loss += loss.item()
            cost_list.append(running_loss)
            if i % 2000 == 1999:    # print every 2000 mini-batches
                print('[%d, %5d] loss: %.3f' %
                      (epoch + 1, i + 1, running_loss / 2000))
                running_loss = 0.0
            # 7) Track accuracy
    #         for x_test, y_test in validation_loader:

    #             z=net(x_test)
    #             _,yhat=torch.max(z.data,1)

```

```
36      #             correct+=(yhat==y_test).sum().item()
37
38
39      #         accuracy=correct/N_test
40
41      #         accuracy_list.append(accuracy)
42      print('Finished Training')
43      torch.save(net.state_dict(), './trained_CNN')
```

FUNCTION B.3.: The function to train the CNN.

```
1       filename = '/path/to/video/file.avi'
2       csv_path = '/path/to/classificationVector.csv'
3
4       transformed_dataset = MotionEnergyDataset(csv_path,
5                                                 filename,
6                                                 transform = transforms.Compose([
7                                                 _2PILImage(),
8                                                 Rescale(128),
9                                                 _2Tensor()
10                                                ]))
11
12
13      epochs = 100
14      trainloader = torch.utils.data.DataLoader(transformed_dataset, batch_size=1,
15                                                  batch_sampler=None)
16      no_clusters = pd.read_csv(csv_path)
17      no_clusters = max(no_clusters['torch'])
18      net = Net(no_clusters+1)
19      optimizer = torch.optim.Adam(net.parameters(), lr=0.001)
20      train_net(epochs, trainloader, optimizer, net)
```

FUNCTION B.4.: Script loading CNN, dataloader, and the training function. Outputs a trained CNN and saves it automatically

C. Optical Flow Implementation

An alternative, but at this point no longer pursued, approach to extract face movement information is the use of the Farnebäck optical flow algorithm. The aim is to find those pixels whose grayscale value varies the most over the duration of the video. Those pixels should belong to the face regions that move the most and thus convey the most information. Once those pixels are identified their location can be passed to DLC. DLC should be able to track those pixels, assuming that they were identified reliably by the optical flow method. The main steps of the algorithm are quoted below.

FUNCTION C.1 represents our implementation of the Farnebäck algorithm, implemented in MATLAB. It is supposed to be run after the `deeplabcut.extract_frames` function, to replace the `deeplabcut.label_frames` step. Thus, a number of representative randomly selected frames have been extracted by DLC and saved as .png files in a specified folder. `pixel_displacements` takes as arguments both the path to the original videos as well as to the .png frames. It iterates through the video, computing the dense optical flow for each pixel at every step. From these resulting 'velocity vectors' mean magnitude and variance are calculated on-the-go. `largest_variances` FUNCTION C.2 then extracts the `k` pixel locations with the largest variance. Last, `format_as_labelled_frames` FUNCTION C.3 translates these indices to a .csv file that can be read by the next steps of the DLC pipeline. FUNCTION C.4 is a small helper function that parses the file names generated by `deeplabcut.label_frames`.

```
1   function [foi_pixel_location, current_var, current_mean] = ...
2                                       pixel_displacements(video_path, png_path)
3   %% Get 1) pixel displacement values of all pixels for frames selected by
4   %%dlc.extract_frames and 2) the corresponding mean and variance values
5
6   % Read in video
7   vidReader = VideoReader(video_path,'CurrentTime',11);
8
9   % Specify the optical flow estimation method as opticalFlowFarneback.
10  % The output
```

```
11      % is an object specifying the optical flow estimation method and its
12      % properties.
13      opticFlow = opticalFlowFarneback('NumPyramidLevels', 3, 'NumIterations', ...
14          10, 'NeighborhoodSize', 7, 'FilterSize', 10);
15
16      % Input: list of indices of the frames extracted by dlc.extract_frames
17      files = dir(fullfile(png_path, '*.png'));
18      for i=1:length(files)
19          [~, nam] = fileparts(files(i).name);
20          list_of_extracted_frame_indices(i) = sscanf(nam, 'img%d');
21      end
22      clear files nam
23      no_frames = length(list_of_extracted_frame_indices);
24
25      % Initialize loop variables
26      frame_size = [vidReader.Height vidReader.Width];
27      foi_pixel_location = zeros(frame_size(1), frame_size(2)*2, no_frames);
28      delta_x = zeros(frame_size(1), frame_size(2));
29      delta_y = zeros(frame_size(1), frame_size(2));
30      iterator = 1;
31
32      mean_var = zeros(frame_size(1), frame_size(2),3);
33      % Explanation of the meaning of the levels of the matrix
34      new_value = mean_var(:,:,1);
35      current_mean = mean_var(:,:,2);
36      current_var = mean_var(:,:,3);
37      frame = 1;
38      while hasFrame(vidReader)
39
40          % Part 1: get pixel values of all frames
41          iterator = iterator + 1;
42          % Read image frames from the VideoReader object and convert to grayscale
43          % images. Estimate the optical flow from consecutive image frames.
44          frameRGB = readFrame(vidReader);
45          frameGray = frameRGB;%rgb2gray(frameRGB);
46          flow = estimateFlow(opticFlow,frameGray);
47
48          % Continuously run a cumsum of the pixel displacements in x and y
49          % coordinates (i.e. a cumsum of flow.Vx and flow.Vy, respectively)
50          delta_x = delta_x + flow.Vx;
51          delta_y = delta_y + flow.Vy;
```

```
52
53          % To obtain the pixel locations at timepoint t= frame_number, save
54          % the cumsum at this point
55          if ismember(iterator, list_of_extracted_frame_indices)
56              i = find(list_of_extracted_frame_indices == iterator);
57
58              i
59              foi_pixel_location(:,1:2:end, i) = delta_x;
60              foi_pixel_location(:,2:2:end, i) = delta_y;
61          end
62
63          % Part 2: get variance and mean of magnitude values
64          new_value = flow.Magnitude(); %replace the 'frame'
65          % calculate mean
66          temp_mean = current_mean + (new_value - current_mean)/ frame;
67          % calculate variance
68          if frame == 1
69              temp_var = zeros(frame_size(1), frame_size(2));
70          elseif frame ~= 1
71              temp_var = ((frame - 2).*current_var + (frame - 1).* ...
72                  (current_mean - temp_mean).^2 + (new_value - temp_mean).^2)./(frame - 1);
73          end
74          % Variance: iteratively update
75          current_var = temp_var;
76          % Mean: iteratively update
77          current_mean = temp_mean;
78
79
80          frame = frame + 1;
81      end
82      variance_array = current_var;
83      mean_array = current_mean;
84  end
```

FUNCTION C.1.: Extract mean pixel displacements and their variances.

```
1  function index = largest_variances(k, variance_array)
2      %% Get the pixel locations of the k largest variances obtained from optical flow
3
4      k_largest = reshape(variance_array, [], 1);
5      k_largest = sort(k_largest, 'descend');
```

```
    index = k_largest;
    % Here it was possible to select a range other than simply the N largest
    % values (which ends up scoring whiskers only)
    % Now subsetting only value range within heuristically determined
    % bounds (239.1583 to 3.3989e+04)
%     temp_ind = find(k_largest < 3.3989e+04 & k_largest > 239.158);
%     variance_roi = k_largest(temp_ind(1):temp_ind(end));
%     % Now getting the k largest of the selected value region
%     k_largest = variance_roi(1:k, 1);

    % This returns the linear indices of those 200 largest variances
    index = zeros(k, 1);
    for i=1:k
        index(i) = find(variance_array==k_largest(i));
    end
    % Convert to indices matching the array size
    [index(1:k,1), index(1:k,2)] = ind2sub([size(variance_array,1), ...
                                            size(variance_array,2)], index);
end
```

FUNCTION C.2.: Obtain the indices to the pixels with the largest overall variance. Other criteria might be used at this point, and example is commented out.

```
function format_as_labelled_frames(png_path, variance_array, displacement_array, ...
    selected_pixel_locations, experimenter_name)
    %% This takes the pixel location information extracted by
    %% pixel_displacements.m and casts it into a .csv that dlc can work with
    %Create column labels
    header{1,1} = {'scorer'};
    header{2,1} = {'bodyparts'};
    header{3,1} = {'coords'};
    for i=2:size(variance_array, 2) + 1
        header{1, i} = {experimenter_name};
        header{2, i} = {['Variable_', num2str(i)]};
        selected_pixel_locations_x = [2:2:size(variance_array, 2) + 1];
        selected_pixel_locations_y = [3:2:size(variance_array, 2) + 2];
        header{3, selected_pixel_locations_x(fix(i/2))} = 'x';
        header{3, selected_pixel_locations_y(fix(i/2))} = 'y';
    end

    % Create row labels
```

```
    [file_info, ~, file_number] = get_png_info(png_path);

    row_names = cell(file_number, 1);

    for i=1:file_number
        row_names{i,1} = [file_info(i).folder, file_info(i).name];
        for j=2:size(variance_array,2) + 1
            row_names{i, j} = 0;
        end
    end
    % clear file_info file_number

    %Convert to table
    output_csv = [header; row_names];
    output_csv = cell2table(output_csv);
    clear header

    % Paste in data
    for i=1:file_number
        for j=1:size(selected_pixel_locations,1)
            selected_pixel_locations_xy = j + 1;
            slice = {displacement_array(selected_pixel_locations(j,1), ...
                selected_pixel_locations(j,2), i)};
            output_csv{i + 3, ...
                selected_pixel_locations_xy:selected_pixel_locations_xy + 1} = ...
                slice;
        end
    end

    % Write out
    writetable(output_csv,[png_path filesep 'output.csv'], 'WriteVariableNames',...
        false, 'WriteRowNames', true);
end
```

FUNCTION C.3.: Output the location of the pixels selected by FUNCTION C.2. This output is automatically formatted to the requirements posed by DLC.

```
    function [file_info, file_names, no_files] = get_png_info(path)
        %% Read file names and index numbers from png image folder generated by
        %% dlc.extract_frames
        % Input: list of indices of the frames extracted by dlc.extract_frames
```

```
5           file_info = dir(fullfile(path, '*.png'));
6           for i=1:length(file_info)
7               [~, file_names] = fileparts(file_info(i).name);
8               list_of_extracted_frame_indices(i) = sscanf(file_names, 'img%d');
9           end
10          no_files = length(list_of_extracted_frame_indices);
11      end
```

FUNCTION C.4.: A small utility function extracting the file names of the .png files generated by DLC.

D. Histogram of Oriented Gradients calculation

HOG calculation is a simple process extracting first-order features from a given image using linear algebra. The image is first separated into squares of length equal to a set `CellSize` FIGURE D.1 (a). Gradient-detection kernels are then applied to each cell in both dimensions. Today, the Sobel kernel is most commonly used [97], since it detects and focusses edges FIGURE D.1 (b). The resulting vector field is shown in FIGURE D.1 (c). These vectors are defined by a magnitude and a direction, which can be calculated using the formulas indicated in FIGURE D.1 (b). In the next step, the histograms are computed from the resulting matrices FIGURE D.1 (d). Bin edges are predetermined by specifying `UsedSignOrientation`, meaning whether to bin into half- or full-circle orientations (180° and 360°, respectively). Based on the selection n bins are generated equally spaced between 0 and the maximum angle. The magnitude of each gradient vector is binned based on its matching orientation angle into the HOG vector FIGURE D.1 (e). In our case, those cell-HOG vectors are simply concatenated into a global image HOG, but oftentimes multiple cell-HOG vectors are first combined into a block-HOG vector, a block being the summary of a predefined number of cells. If used, blocks are normalized to account for regional lighting differences that might affect local gradient magnitudes. The result, either way, is a low-dimensional vector description of the image's edges and their orientations. HOG features have low computational demands because of their underlying simple linear algebra. Notably, they are not affine-invariant, meaning that they perform poorly in extracting the same features viewed from a different angle.

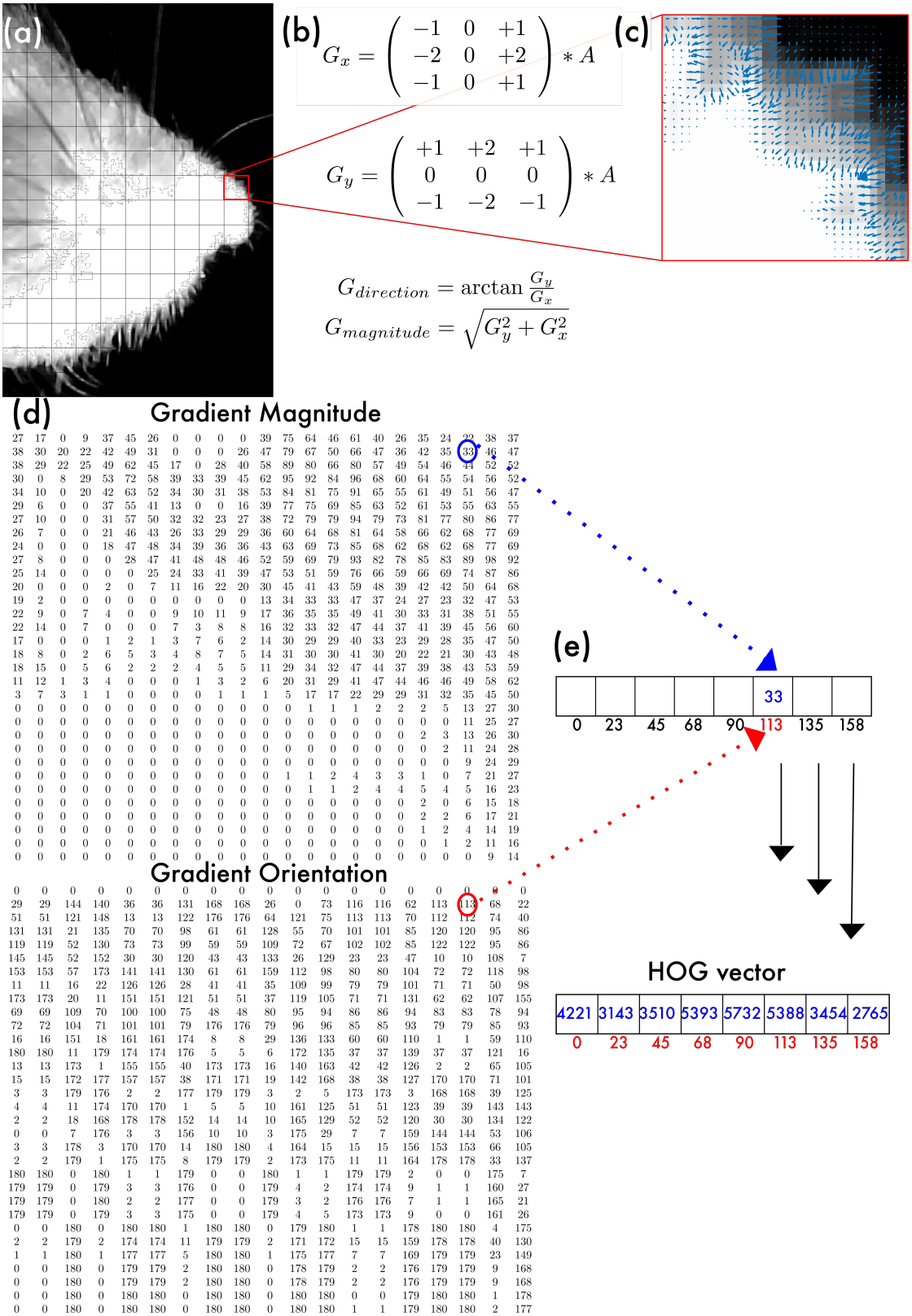

FIGURE D.1.: Outline of the HOG vector generation. (a) Each image is subdivided into cells, which are processed individually. (b) The *Sobel* kernel is convoluted with each cell, performing smoothing and derivation. Vector magnitude and orientation are calculated using the formulas indicated. (c) Gradient vector field of the exemplary cell, vectors are scaled by a factor 1.5. (d) Gradient magnitude and -orientation results for each pixel of exemplary cell. Values are rounded to the nearest integer. (e) Binning of the gradient magnitudes by their orientation. (d) The resulting histogram for the cell.

List of Figures

Bibliography

[1] S Abd-El-Malek. "The part played by the tongue in mastication and deglutition." In: *Journal of anatomy* 89.2 (1955), pp. 250–4. ISSN: 0021-8782.

[2] Ralph Adolphs. "How should neuroscience study emotions? By distinguishing emotion states, concepts, and experiences". In: *Social Cognitive and Affective Neuroscience* 12.1 (2017), pp. 24–31. ISSN: 17495024. DOI: 10.1093/scan/nsw153.

[3] Ralph Adolphs. "Reply to Barrett: affective neuroscience needs objective criteria for emotions". In: *Social cognitive and affective neuroscience* 12.1 (2017), pp. 32–33. ISSN: 17495024. DOI: 10.1093/scan/nsw155.

[4] Jeanne Altmann. "Observational study of behavior: sampling methods". In: *Behaviour* 49.3-4 (1974), pp. 227–266.

[5] Lisa Feldman Barrett. "Was darwin wrong about emotional expressions?" In: *Current Directions in Psychological Science* 20.6 (2011), pp. 400–406. ISSN: 09637214. DOI: 10.1177/0963721411429125.

[6] Lisa Feldman Barrett. "Functionalism cannot save the classical view of emotion". In: *Social Cognitive and Affective Neuroscience* 12.1 (2017), pp. 34–36. ISSN: 17495024. DOI: 10.1093/scan/nsw156.

[7] Steven S Beauchemin and John L Barron. "The computation of optical flow". In: *ACM computing surveys (CSUR)* 27.3 (1995), pp. 433–466.

[8] Gershon Berkson. "Development of abnormal setereotyped behaviors". In: *Developmental Psychobiology: The Journal of the International Society for Developmental Psychobiology* 1.2 (1968), pp. 118–132.

[9] Gordon J Berman et al. "Mapping the stereotyped behaviour of freely moving fruit flies". In: (2014).

[10] Ben Bland. *Sensor Fusion: The Only Way to Measure True Emotion*. 2017. URL: https://sensum.co/blog/sensor-fusion-the-only-way-to-measure-true-emotion (visited on 07/24/2020).

[11] Ben Bland. *AI Ethics – From Principles to Standards*. 2019. URL: https://sensum.co/blog/ai-ethics-from-principles-to-standards (visited on 07/24/2020).

[12] Alexander Borst and Frédéric E. Theunissen. "Information Theory and Neural Coding". In: *North-Holland Mathematical Library* 51.C (1993), pp. 307–340. ISSN: 09246509. DOI: 10.1016/S0924-6509(08)70042-4.

[13] M Brecht et al. "Movements evoked by intracellular stimulation of single pyramidal cells in layer 5 and 6 of rat motor cortex". In: *Nature* 427.6976 (2003), pp. 704–710.

[14] Jerí L. Bryant et al. "Cerebellar cortical output encodes temporal aspects of rhythmic licking movements and is necessary for normal licking frequency". In: *European Journal of Neuroscience* 32.1 (2010), pp. 41–52. ISSN: 0953816X. DOI: 10.1111/j.1460-9568.2010.07244.x.

[15] Cátia C Caeiro, Anne M Burrows, and Bridget M Waller. "Development and application of CatFACS: Are human cat adopters influenced by cat facial expressions?" In: *Applied Animal Behaviour Science* 189 (2017), pp. 66–78.

[16] Cátia C Caeiro et al. "OrangFACS: A muscle-based facial movement coding system for orangutans (Pongo spp.)" In: *International Journal of Primatology* 34.1 (2013), pp. 115–129.

[17] Daniel Canedo and António J.R. Neves. "Facial expression recognition using computer vision: A systematic review". In: *Applied Sciences (Switzerland)* 9.21 (2019), pp. 1–31. ISSN: 20763417. DOI: 10.3390/app9214678.

[18] John Canny. "A computational approach to edge detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[19] Alan Carleton, Riccardo Accolla, and Sidney A. Simon. "Coding in the mammalian gustatory system". In: *Trends in Neurosciences* 33.7 (2010), pp. 326–334. ISSN: 01662236. DOI: 10.1016/j.tins.2010.04.002.

[20] James M Carroll and James A Russell. "Do facial expressions signal specific emotions? Judging emotion from the face in context." In: *Journal of personality and social psychology* 70.2 (1996), p. 205.

[21] George E. Carvell et al. "Electromyographic Activity of Mystacial Pad Musculature during Whisking Behavior in the Rat". In: *Somatosensory & Motor Research* 8.2 (1991), pp. 159–164. DOI: 10.3109/08990229109144740.

[22] S. Chakrabarti and C. Schwarz. "Studying motor cortex function using the rodent vibrissal system". In: *e-Neuroforum* 5.1 (2014), pp. 20–27. ISSN: 0947-0875. DOI: 10.1007/s13295-014-0051-y.

[23] A. Chastel. *Leonardo on Art and the Artist*. Dover Fine Art, History of Art Series. Dover Publications, 2002. ISBN: 9780486421667.

[24] Sasank Chilamkurthy. *Writing Custom Datasets, DataLoaders and TransformsMS*. 2017. URL: https://pytorch.org/tutorials/beginner/data_loading_tutorial.html (visited on 07/15/2020).

[25] Charles K Chui and Qingtang Jiang. "Wavelet analysis". In: *Applied Mathematics*. Springer, 2013, pp. 499–546.

[26] Andrzej Cichocki et al. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. 2009, pp. 1–477. ISBN: 9780470746660. DOI: 10.1002/9780470747278.

[27] Jason R Climer, Daniel A Dombeck, and D Ph. "Information theoretic approaches to deciphering the neural code with functional fluorescence imaging". In: Mi (2020).

[28] Wikipedia Contributors. *Motor Homunculus*. 2020. URL: https://commons.wikimedia.org/wiki/Category:Cortical_homunculus#/media/File:Motor_homunculus.svg (visited on 07/24/2020).

[29] Wikipedia Contributors. *Sensory and motor homunculi*. 2020. URL: https://commons.wikimedia.org/wiki/Category:Cortical_homunculus#/media/File:Sensory_and_motor_homunculi.jpg (visited on 07/24/2020).

[30] Harihara Santosh Dadi and GK Mohan Pillutla. "Improved face recognition rate using HOG features and SVM classifier". In: *IOSR Journal of Electronics and Communication Engineering* 11.4 (2016), pp. 34–44.

[31] Mohamed Dahmane and Jean Meunier. "Emotion recognition using dynamic grid-based HoG features". In: *Face and Gesture 2011*. IEEE. 2011, pp. 884–888.

[32] Charles Darwin. *The expression of emotions in man and animals*. 1872, pp. 183–204. DOI: 10.1037/10025-007.

[33] Craig Dunn David Britch. *Perceived Emotion Recognition Using the Face API*. 2018. URL: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/azure-cognitive-services/emotion-recognition (visited on 07/24/2020).

[34] E Roy Davies. *Computer vision: principles, algorithms, applications, learning*. Academic Press, 2017.

[35] Erwin B. Defensor et al. "Facial expressions of mice in aggressive and fearful contexts". In: *Physiology and Behavior* 107.5 (2012), pp. 680–685. ISSN: 00319384. DOI: 10.1016/j.physbeh.2012.03.024.

[36] Martin Deschênes et al. "Inhibition, not excitation, drives rhythmic whisking". In: *Neuron* 90.2 (2016), pp. 374–387.

[37] Nejc Dolensek et al. "Facial expressions of emotion states and their neuronal correlates in mice". In: 94.April (2020), pp. 89–94.

[38] Christian L. Ebbesen et al. "More than just a "motor": Recent surprises from the frontal cortex". In: *Journal of Neuroscience* 38.44 (2018), pp. 9402–9413. ISSN: 15292401. DOI: 10.1523/JNEUROSCI.1671-18.2018.

[39] Christian L Ebbesen and Robert C Froemke. "Automatic tracking of mouse social posture dynamics by 3D videography, deep learning and GPU-accelerated robust optimization". In: *bioRxiv* (2020).

[40] Friesen W. V. Ekman P. "Constants across cultures in the face and emotion." In: *Journal of Personality and Social Psychology* 17.2 (1971), p. 124.

[41] Paul Ekman. "An Argument for Basic Emotions". In: *Cognition and Emotion* 6.3-4 (1992), pp. 169–200. ISSN: 14640600. DOI: 10.1080/02699939208411068.

[42] Paul Ekman and Daniel Cordaro. "What is meant by calling emotions basic". In: *Emotion Review* 3.4 (2011), pp. 364–370. ISSN: 17540739. DOI: 10.1177/1754073911410740.

[43] eyeris.ai. *Human Behavior Understanding*. 2020. URL: https://www.eyeris.ai/technology/ (visited on 07/24/2020).

[44] Gunnar Farnebäck. "Two-frame motion estimation based on polynomial expansion". In: *Scandinavian conference on Image analysis*. Springer. 2003, pp. 363–370.

[45] D Ferrier. "Experiments on the brains of monkeys. London". In: *Proc. R. Soc.* 1875.

[46] Kathryn Finlayson et al. "Facial indicators of positive emotions in rats". In: *PloS one* 11.11 (2016), e0166446.

[47] G Fritsch. "Uber die elektrische Erregbarkeit des Grosshirns". In: *Arch, anat. Physiol. Wiss. Med.* 37 (1870), pp. 300–332.

[48] Y. Furuta et al. "Herpes simplex virus type 1 reactivation and antiviral therapy in patients with acute peripheral facial palsy". In: *Auris Nasus Larynx* 28.SUPPL. (2001), pp. 3–7. ISSN: 03858146. DOI: 10.1016/S0385-8146(00)00105-X.

[49] Puhong Gao et al. "Whisker motor cortex ablation and whisker movement patterns". In: *Somatosensory and Motor Research* 20.3-4 (2003), pp. 191–198. ISSN: 08990220. DOI: 10.1080/08990220310001622924.

[50] Augusta Gaspar and Francisco G Esteves. "Preschooler's faces in spontaneous emotional contexts—how well do they match adult facial expression prototypes?" In: *International Journal of Behavioral Development* 36.5 (2012), pp. 348–357.

[51] By Benoit Girard and Camilla Bellone. "Revealing animal emotions". In: 368.6486 (2020).

[52] Jacob M Graving et al. "DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning". In: *Elife* 8 (2019), e47994.

[53] Valery Grinevich, Michael Brecht, and Pavel Osten. "Monosynaptic pathway from rat vibrissa motor cortex to facial motor neurons revealed by lentivirus-based axonal tracing". In: *Journal of Neuroscience* 25.36 (2005), pp. 8250–8258. ISSN: 02706474. DOI: 10.1523/JNEUROSCI.2235-05.2005.

[54] Jason M. Guest et al. "3D reconstruction and standardization of the rat facial nucleus for precise mapping of vibrissal motor networks". In: *Neuroscience* 368 (2018), pp. 171–186. ISSN: 18737544. DOI: 10.1016/j.neuroscience.2017.09.031.

[55] Zengcai V Guo et al. "Procedures for behavioral experiments in head-fixed mice". In: *PloS one* 9.2 (2014), e88678.

[56] Zengcai V Guo et al. "Flow of cortical activity underlying a tactile decision in mice". In: *Neuron* 81.1 (2014), pp. 179–194.

[57] James V. Haxby, Andrew C. Connolly, and J. Swaroop Guntupalli. "Decoding Neural Representational Spaces Using Multivariate Pattern Analysis". In: *Annual Review of Neuroscience* 37.1 (2014), pp. 435–456. ISSN: 0147-006X. DOI: 10.1146/annurev-neuro-062012-170325.

[58] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[59] Douglas Heaven. "Why faces don't always tell the truth about feelings." In: *Nature* 578.7796 (2020), pp. 502–504.

[60] Anthony Holtmaat and Karel Svoboda. "Experience-dependent structural synaptic plasticity in the mammalian brain". In: *Nature Reviews Neuroscience* 10.9 (2009), pp. 647–658.

[61] Eldar Insafutdinov et al. "Deepercut: A deeper, stronger, and faster multi-person pose estimation model". In: *European Conference on Computer Vision*. Springer. 2016, pp. 34–50.

[62] Carroll E Izard. "Innate and universal facial expressions: evidence from developmental and cross-cultural research." In: (1994).

[63] Yajie Jiang et al. "Fast Non-Negative Tensor Factorization based on CP decomposition for 3D Facial Expression Recognition". In: *International Conference on Signal Processing Proceedings, ICSP* 2018-Augus (2019), pp. 527–532. DOI: 10.1109/ICSP.2018.8652345.

[64] J. P. Jones and L. A. Palmer. "The two-dimensional spatial structure of simple receptive fields in cat striate cortex". In: *Journal of Neurophysiology* 58.6 (1987), pp. 1187–1211. ISSN: 00223077. DOI: 10.1152/jn.1987.58.6.1187.

[65] David Kleinfeld et al. "The brainstem oscillator for whisking and the case for breathing as the master clock for orofacial motor actions". In: *Cold Spring Harbor Symposia on Quantitative Biology* 79 (2014), pp. 29–39. ISSN: 19434456. DOI: 10.1101/sqb.2014.79.024794.

[66] Takaki Komiyama et al. "Learning-related fine-scale specificity imaged in motor cortex circuits of behaving mice". In: *Nature* 464.7292 (2010), pp. 1182–1186. ISSN: 00280836. DOI: 10.1038/nature08897.

[67] Anastasia Kurnikova et al. "Coordination of Orofacial Motor Actions into Exploratory Behavior by Rat". In: *Current Biology* 27.5 (2017), pp. 688–696. ISSN: 09609822. DOI: 10.1016/j.cub.2017.01.013.

[68] Dale J. Langford et al. "Coding of facial expressions of pain in the laboratory mouse". In: *Nature Methods* 7.6 (2010), pp. 447–449. ISSN: 15487091. DOI: 10.1038/nmeth.1455.

[69] Nuo Li et al. "A motor cortex circuit for motor planning and movement". In: *Nature* 519.7541 (2015), pp. 51–56.

[70] Nuo Li et al. "Robust neuronal dynamics in premotor cortex during motor planning". In: *Nature* 532.7600 (2016), pp. 459–464.

[71] C Lin. *Introduction to Motion Estimation with Optical Flow*. 2019. URL: https://nanonets.com/blog/optical-flow.

[72] Longping Liu, Robin Arnold, and Marcus Robinson. "Dissection and Exposure of the Whole Course of Deep Nerves in Human Head Specimens after Decalcification". In: *International Journal of Otolaryngology* 2012 (2012), pp. 1–7. ISSN: 1687-9201. DOI: 10.1155/2012/418650.

[73] David G Lowe et al. "Object recognition from local scale-invariant features." In: *iccv*. Vol. 99. 2. 1999, pp. 1150–1157.

[74] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[75] Alexander Mathis et al. "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning". In: *Nature neuroscience* 21.9 (2018), pp. 1281–1289.

[76] Mackenzie Weygandt Mathis and Alexander Mathis. "Deep learning tools for the measurement of animal behavior in neuroscience". In: *Current Opinion in Neurobiology* 60 (2020), pp. 1–11. ISSN: 18736882. DOI: 10.1016/j.conb.2019.10.008.

[77] Johannes M. Mayrhofer et al. "Distinct Contributions of Whisker Sensory Cortex and Tongue-Jaw Motor Cortex in a Goal-Directed Sensorimotor Transformation". In: *Neuron* 103.6 (2019), 1034–1043.e5. ISSN: 10974199. DOI: 10.1016/j.neuron.2019.07.008.

[78] Daniel McDuff et al. "Predicting ad liking and purchase intent: Large-scale analysis of facial responses to ads". In: *IEEE Transactions on Affective Computing* 6.3 (2014), pp. 223–235.

[79] Daniel McDuff et al. "AFFDEX SDK: a cross-platform real-time multi-face expression recognition toolkit". In: *Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems*. 2016, pp. 3723–3726.

[80] Lauren E. McElvain et al. "Circuits in the rodent brainstem that control whisking in concert with other orofacial motor actions". In: *Neuroscience* 368 (2018), pp. 152–170. ISSN: 18737544. DOI: 10.1016/j.neuroscience.2017.08.034.

[81] Nicole Mercer Lindsay et al. "Orofacial Movements Involve Parallel Corticobulbar Projections from Motor Cortex to Trigeminal Premotor Nuclei". In: *Neuron* 104.4 (2019), 765–780.e3. ISSN: 10974199. DOI: 10.1016/j.neuron.2019.08.032.

[82] Jeffrey D. Moore et al. "Hierarchy of orofacial rhythms revealed through whisking and breathing". In: *Nature* 497.7448 (2013), pp. 205–210. ISSN: 00280836. DOI: 10.1038/nature12076.

[83] Simon Musall et al. "Harnessing behavioral diversity to understand neural computations for cognition". In: *Current opinion in neurobiology* 58 (2019), pp. 229–238.

[84] Simon Musall et al. "Single-trial neural dynamics are dominated by richly varied movements". In: *Nature neuroscience* 22.10 (2019), pp. 1677–1686.

[85] Tanmay Nath et al. "Using DeepLabCut for 3D markerless pose estimation across species and behaviors". In: *Nature protocols* 14.7 (2019), pp. 2152–2176.

[86] Nemesysco. *Research*. 2020. URL: https://www.nemesysco.com/research/ (visited on 07/24/2020).

[87] Lisa A Parr et al. "Brief communication: MaqFACS: a muscle-based facial movement coding system for the rhesus macaque". In: *American journal of physical anthropology* 143.4 (2010), pp. 625–630.

[88] Friedrich Paulsen and Jens Waschke. *Sobotta, Atlas der Anatomie des Menschen Band 3: Kopf, Hals und Neuroanatomie*. Elsevier Health Sciences, 2011.

[89] Wilder Penfield and Theodore Rasmussen. "The cerebral cortex of man; a clinical study of localization of function." In: (1950).

[90] Talmo D Pereira et al. "Fast animal pose estimation using deep neural networks". In: *Nature methods* 16.1 (2019), pp. 117–125.

[91] Andrew J. Peters, Haixin Liu, and Takaki Komiyama. "Learning in the Rodent Motor Cortex". In: *Annual Review of Neuroscience* 40.1 (2017), pp. 77–97. ISSN: 0147-006X. DOI: 10.1146/annurev-neuro-072116-031407.

[92] Robert Prevedel et al. "Fast volumetric calcium imaging across multiple cortical layers using sculpted light". In: *Nature methods* 13.12 (2016), pp. 1021–1028.

[93] James A Russell. "Is there universal recognition of emotion from facial expression? A review of the cross-cultural studies." In: *Psychological bulletin* 115.1 (1994), p. 102.

[94] Andrew B. Schwartz. "Useful signals from motor cortex". In: *Journal of Physiology* 579.3 (2007), pp. 581–601. ISSN: 00223751. DOI: 10.1113/jphysiol.2006.126698.

[95] Erika H Siegel et al. "Emotion fingerprints or emotion populations? A meta-analytic investigation of autonomic features of emotion categories." In: *Psychological bulletin* 144.4 (2018), p. 343.

[96] Jared B. Smith et al. "Corticofugal projection patterns of whisker sensorimotor cortex to the sensory trigeminal nuclei". In: *Frontiers in Neural Circuits* 9.September (2015), pp. 1–14. ISSN: 16625110. DOI: 10.3389/fncir.2015.00053.

[97] Irwin Sobel and Gary Feldman. "A 3x3 isotropic gradient operator for image processing". In: *a talk at the Stanford Artificial Project in* (1968), pp. 271–272.

[98] Charles F. Sonntag. "The Comparative Anatomy of the Tongues of the Mammalia.— VIII. Carnivora". In: *Proceedings of the Zoological Society of London* 93.1 (1923), pp. 129–153. ISSN: 14697998. DOI: 10.1111/j.1096-3642.1923.tb02177.x.

[99] Susana G. Sotocinal et al. "The Rat Grimace Scale: A partially automated method for quantifying pain in the laboratory rat via facial expressions". In: *Molecular Pain* 7 (2011), pp. 1–10. ISSN: 17448069. DOI: 10.1186/1744-8069-7-55.

[100] Varun Sreenivasan et al. "Parallel pathways from motor and somatosensory cortex for controlling whisker movements in mice". In: *European Journal of Neuroscience* 41.3 (2015), pp. 354–367. ISSN: 14609568. DOI: 10.1111/ejn.12800.

[101] Carsen Stringer et al. "Spontaneous behaviors drive multidimensional, brainwide activity". In: 7893 (2019). DOI: 10.1126/science.aav7893.

[102] Hengliang Tan, Bing Yang, and Zhengming Ma. "Face recognition based on the fusion of global and local HOG features of face images". In: *IET computer vision* 8.3 (2013), pp. 224–234.

[103] Visage Technologies. *Emotion Recognition*. 2020. URL: https://visagetechnologies.com/emotion-recognition (visited on 07/24/2020).

[104] S. S. Tomkins and R. McCarter. "What and Where Are the Primary Affects? Some Evidence for a Theory." In: *Perceptual and motor skills* 18 (1964), pp. 119–158. ISSN: 00315125. DOI: 10.2466/pms.1964.18.1.119.

[105] Martin Trepel. *Neuroanatomie: Struktur und Funktion*. Vol. 3. Urban & Fischer, 2003.

[106] Angus Trumble. *A brief history of the smile.* Basic Books, 2004.

[107] Levi Valgaerts et al. "Dense versus sparse approaches for estimating the fundamental matrix". In: *International Journal of Computer Vision* 96.2 (2012), pp. 212–234.

[108] Sarah-Jane Vick et al. "A cross-species comparison of facial morphology and movement in humans and chimpanzees using the facial action coding system (FACS)". In: *Journal of nonverbal behavior* 31.1 (2007), pp. 1–20.

[109] Paul Viola and Michael Jones. "Robust Real-time Object Detection". In: *International Journal of Computer Vision*. 2001.

[110] Bridget M Waller et al. "GibbonFACS: a muscle-based facial movement coding system for hylobatids". In: *International Journal of Primatology* 33.4 (2012), pp. 809–821.

[111] Bridget M Waller et al. "Paedomorphic facial expressions give dogs a selective advantage". In: *PLoS one* 8.12 (2013), e82686.

[112] Guiming Wang. "Machine learning for inferring animal behavior from location and movement data". In: *Ecological Informatics* 49.August 2018 (2019), pp. 69–76. ISSN: 15749541. DOI: 10.1016/j.ecoinf.2018.12.002.

[113] Jen Wathan et al. "EquiFACS: the equine facial action coding system". In: *PLoS one* 10.8 (2015), e0131738.

[114] Quentin Welniarz, Isabelle Dusart, and Emmanuel Roze. "The corticospinal tract: Evolution, development, and human disorders". In: *Developmental Neurobiology* 77.7 (2017), pp. 810–829. ISSN: 1932846X. DOI: 10.1002/dneu.22455.

[115] Hans Welzl and Jan Bureš. "Lick-synchronized breathing in rats". In: *Physiology and Behavior* 18.4 (1977), pp. 751–753. ISSN: 00319384. DOI: 10.1016/0031-9384(77)90079-8.

[116] Zuzsanna Wiesenfeld. "Licking Behavior: Evidence of Hypoglossal Oscillator". In: 824.1976 (1977), pp. 1976–1978.

[117] Michal Roth William T. Freeman. *Orientation Histograms for Hand Gesture Recognition*. Tech. rep. TR94-03. Cambridge, MA 02139: MERL - Mitsubishi Electric Research Laboratories, Dec. 1994.

[118] Lawrence E Wineski. "Facial morphology and vibrissal movement in the golden hamster". In: *Journal of Morphology* 183.2 (1985), pp. 199–217.

[119] James W Woods. "Behavior of chronic decerebrate rats". In: *Journal of Neurophysiology* 27.4 (1964), pp. 635–644.

[120] Jingdong Zhang, Pifu Luo, and William W. Pendlebury. "Light and electron microscopic observations of a direct projection from mesencephalic trigeminal nucleus neurons to hypoglossal motoneurons in the rat". In: *Brain Research* 917.1 (2001), pp. 67–80. ISSN: 00068993. DOI: 10.1016/S0006-8993(01)02911-0.

[121] Christian Zimmermann et al. "FreiPose: A Deep Learning Framework for Precise Animal Motion Capture in 3D Spaces". In: *bioRxiv* (2020).