Management of Electrophysiological Data & Metadata

-

Making complex experiments accessible to yourself and others.

Der Fakultät Mathematik, Informatik und Naturwissenschaften
der RWTH Aachen University vorgelegte Dissertation
zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften

von
Master of Science
(Neuroscience)

Lyuba Zehl

aus Köln

April 2016

List of own papers with authors contributions

The work presented in this thesis is in main parts based on the following four papers which are published or currently in preparation to be published in peer-reviewed international journals.

Massively parallel multi-electrode recordings of macaque motor cortex during an instructed delayed reach-to-grasp task

by Thomas Brochier*,* Lyuba Zehl*, *Yaoyao Hao, Margaux Duret, Julia Sprenger, Michael Denker, Sonja Grün, and Alexa Riehle

Expected submission: end of 2016 / beginning of 2017 (Brochier et al., prep). Authors marked with an * will share first authorship.

The authors had the following contributions:

Thomas Brochier designed, set up and performed the experiment and wrote the manuscript. **Lyuba Zehl** designed and performed the data and metadata management of the experiment, developed and implemented the data and metadata loading and pre-processing routines, wrote the manuscript and designed the corresponding figures. ***Yaayao Hao*** performed the experiment, helped with technical issues of the experimental setup and provided valuable feedback for the manuscript. ***Margaux Duret*** was involved in setting up and performing the experiment and corresponding pre-processing steps, and provided valuable feedback for the manuscript. ***Julia Sprenger*** was involved in implementing experimental pre-processing steps, supported the implementation of the data and metadata loading routines, and provided valuable feedback for the manuscript. ***Michael Denker*** provided valuable feedback for the data and metadata management, was involved in implementing the data and metadata loading routines, and provided valuable feedback for the manuscript. ***Sonja Grün*** was involved in writing the manuscript and provided valuable feedback. ***Alexa Riehle*** was involved in setting up performing the experiment, performed the spike sorting and provided valuable feedback for the manuscript.

Handling Metadata in a Neurophysiology Laboratory

by* Lyuba Zehl, *Florent Jaillet, Adrian Stoewer, Jan Grewe, Andrey Sobolev, Thomas Wachtler, Thomas Brochier, Alexa Riehle, Michael Denker, and Sonja Grün

Submitted (23 March 2016) and accepted (27 June 2016) in Frontiers of Neuroinformatics (Zehl et al., 2016). Preliminary, this paper was presented as a poster at the Neuroinformatics 2014 in Leiden, Netherlands, 25 Aug - 27 Aug, 2014 (Zehl et al., 2014b).

The authors had the following contributions:

Lyuba Zehl reorganized and designed the metadata sources, implemented the metadata management, contributed to the open source project odML, and wrote the manuscript. ***Florent Jaillet*** contributed to the open source project odML and was involved in writing the manuscript. ***Adrian Stoewer*** provided major contributions to the open source project odML and provided valuable feedback for the project and the manuscript. ***Jan Grewe*** provided is one of the original developers and major contributor to the open source project odML and provided valuable feedback for the project and the manuscript. ***Andrey Sobolev*** provided major contributions to the open source project odML and provided valuable feedback for the manuscript. ***Thomas Wachtler*** is one of the original developers of the open source project odML and provided valuable feedback for the project and the manuscript. ***Thomas Brochier*** designed, set up and performed the example experiment, was involved in the reorganization and design process of the metadata sources, and provided valuable feedback for the manuscript. ***Alexa Riehle*** was involved in setting up and performing the example experiment, was involved in the reorganization and design process of the metadata sources, and provided valuable feedback for the manuscript. ***Michael Denker*** contributed to the open source project odML, provided valuable feedback for the project and wrote the manuscript. ***Sonja Grün*** provided valuable feedback for the project and wrote the manuscript.

LFP beta amplitude is predictive of the mesoscopic spatio-temporal phase patterns

by Michael Denker,* Lyuba Zehl, *Bjørg Kilavik, Markus Diesmann, Thomas Brochier, Alexa Riehle, and Sonja Grün

Expected submission: end of 2016 / beginning of 2017 (Denker et al., prep).

The authors had the following contributions:

Michael Denker designed and performed the analysis, and wrote the manuscript. **Lyuba Zehl** designed and performed the analysis and provided feedback for the manuscript. ***Bjørg Kilavik*** was involved in writing the manuscript and provided valuable feedback for the interpretation of the analysis results. ***Markus Diesmann*** provided valuable feedback for the analysis design. ***Thomas Brochier*** designed, set up and performed the example experiment, provided valuable feedback for the interpretation of the analysis results, and was involved in writing the manuscript. ***Alexa Riehle*** was involved in setting up and performing the example experiment, provided valuable feedback for the interpretation of the analysis results, and was involved in writing the manuscript. ***Sonja Grün*** provided valuable feedback for the analysis design and was involved in writing the manuscript.

odMLtables: A user-friendly approach to managing metadata of neurophysiological experiments

by Julia Sprenger*,* Lyuba Zehl*, *Carlos Canova, Jana Pick, Sebastian Bitzenhofer, Joachim Ahlbeck, Kenta Takagaki, Ileana Hanganu-Opatz, Frank Ohl, Sonja Grün, and Michael Denker

(Sprenger et al., prep). Authors marked with an * will share first authorship.

The authors have the following contributions:

Julia Sprenger is writing the manuscript and designed and developed the published software including the GUI. **Lyuba Zehl** implemented the first functionalities of the software, supervised the software design and development including the GUI, and is involved in writing the manuscript. ***Carlos Canova*** is involved in writing the manuscript and provided valuable feedback for the software design. ***Jana Pick*** designed and implemented the first published version of the software including the GUI. ***Sebastian Bitzenhofer, Joachim Ahlbeck, Kenta Takagaki, Ileana Hanganu-Opatz, and Frank Ohl*** are providing valuable feedback for the software usage and the manuscript. ***Sonja Grün*** is providing valuable feedback for manuscript. ***Michael Denker*** was involved in the software design and development and is involved in writing the manuscript.

Summary

As neuroscientists, we are obligated to guarantee the reliability of our research workflows and results. For this reason, most neuroscientific, peer-reviewed journals require, besides a full description of the new scientific findings and used methods, at least also a brief summary of the used data. For a completely open scientific inquiry and to further promote scientific progress as required by most national funding agencies, it would be best to share the raw data along with an analysis paper or independently as a standalone data publication. Unfortunately, especially the neuroscientific community is hesitant to share own research data with third parties, because guidelines, tools and support for the publishing authors to provide data and corresponding adequate information on the experiment are generally hard to formalize and often missing. As a consequence, published scientific results remain unreproducible by other researchers. Along an example of a complex electrophysiological experiment which was conducted by external collaboration partners, I will demonstrate how to share and publish data, but also identify the reasons why researchers, in particular experimentalists in electrophysiology, are hesitant to try the same. For this, I first provide a data descriptor of the experiment following the guidelines of a pure data journal from the Nature publishing group, called Scientific Data. According to the journal guidelines, I describe all information necessary to be able to understand the setup and workflow of the experiment as well as the minimum information necessary to be able to work with the corresponding datasets. The latter requires the provision of a robust data loading routine. To guarantee the access to the data of the experiment, I implemented a commonly usable loading routine for the data formats of the used data acquisition (DAQ) system from Blackrock Microsystems (Cerebus DAQ), and published it as part of an open source data framework, called Neo. Neo has the advantage of representing data in standardized structures that allow researchers to use common analysis routines on different data formats. In addition, it is possible to further annotate the Neo data structures with experiment-specific information on the data. To automatically integrate such information, termed metadata, it is best to have them organised in a machine-readable format. Although several software solutions for such metadata formats exist, they are usually not tested for complex use cases, such as the example experiment. In most cases, they only provide the framework itself as a standardized metadata representation or specification, and no solutions for how to actually compile a useful metadata collection. For the example experiment, I chose a metadata framework, called open metadata Markup Language (odML), an open source project developed by the German Node (G-Node) of the International Neuroinformatics Coordination Facility (INCF). In the second part of the thesis, I demonstrate how to organize metadata for the experiment, to be able to compile and use a corresponding odML metadata collection. To facilitate the compilation process for my collaborators, I developed a Python package, called odMLtables, which facilitates the access to the odML framework by an algorithmic transformation of odML into a spreadsheet format (csv or xls) and back. In addition, I provided a complete workflow for collecting and storing the metadata of the experiment into a comprehensive odML-file collection. Furthermore, I provided a specified data loading routine that automatically annotates the data structures with the corresponding metadata of the collection. The latter improves the workflow in the course of neuroscientific analyses of the data from the example experiment, as demonstrated in the last part of my thesis. In summary, I show that the preparations to properly share research data within a scientific collaboration are cumbersome and time consuming, but essential for successfully publishing data and analysis results for a broader audience of users. To promote data sharing within the neuroscientific community and to provide a better foundation for reproducible research, my thesis offers a coherent strategy for managing electrophysiological data and metadata using a well selected set of available technologies.

Zusammenfassung

Als Neurowissenschaftler sind wir dazu verpflichtet die Zuverlässigkeit und Korrektheit unserer Forschungsarbeit und die damit verbundenen Prozesse und Ergebnisse zu garantieren. Aus diesem Grund verlangen die meisten neurowissenschaftlichen, von Experten begutachteten wissenschaftlichen Zeitschriften neben einer vollständigen Beschreibung der gewonnen wissenschaftlichen Erkenntnisse und der dazu benutzten Methoden zumindest auch eine kurze Zusammenfassung der verwendeten Daten. Viele der nationalen Fördermittelgeber fordern bereits darüber hinaus eine vollständig transparente und wissenschaftlich ausgereifte und wiederverwendbare Dokumentation des wissenschaftlichen Fortschritts. Dies impliziert die Notwendigkeit, die verwendeten Originaldaten gemeinsam mit der Veröffentlichung der wissenschaftlichen Ergebnisse oder davon unabhängig als eigenständige Datenveröffentlichung zu publizieren. Allerdings ist gerade die neurowissenschaftliche Forschungsgemeinde zögerlich, wenn es darum geht die eigenen wissenschaftlichen Daten Dritten zur Verfügung zu stellen. Dies liegt unter anderem daran, dass publizierenden Autoren nicht ausreichend bei der Bereitstellung ihrer Daten und dazugehörigen adäquaten Informationen des Experiments unterstützt werden, da allgemeine Richtlinien schwierig zu formulieren und Werkzeuge komplex und kostenintensiv zu entwickeln sind und daher häufig fehlen. Die daraus resultierende Konsequenz ist, dass publizierte wissenschaftliche Ergebnisse sich nicht von anderen Wissenschaftlern reproduzieren lassen. Entlang des Beispiels eines komplexen elektrophysiologischen Experiments, welches von externen Kooperationspartnern durchgeführt wurde, werde ich demonstrieren wie Daten mit externen Wissenschaftlern geteilt und publiziert werden können. Dabei werde ich die Gründe identifizieren warum Wissenschaftler, und zwar insbesondere Experimentatoren aus der Elektrophysiologie, so zögerlich sind einem solchen Beispiel zu folgen. Dafür präsentiere ich zunächst einen Datendeskriptor dieses Experiments, der den Richtlinien eines reinen Daten Journals der *Nature* Verlagsgruppe mit dem Namen *Scientific Data* entspricht. Übereinstimmend mit den Richtlinien des Journals beschreibe ich alle Informationen, die nötig sind um den Versuchsaufbau und den Versuchsablauf des Experiments zu reproduzieren. Hiermit ist weiterhin die Beschreibung der Mindestinformationen verbunden, welche zur Wiederverwendung der zugehörigen Datensätze notwendig sind. Letzteres setzt die Bereitstellung stabiler Laderoutinen für diese Daten voraus. Um Zugang zu den Daten des Beispiel-Experiments zu garantieren, habe ich deshalb eine gemeinhin nutzbare Laderoutine für die Datenformate des Datenakquisitionssystems (DAQ Systems) von *Blackrock Microsystems* (*Cerebus* DAQ System) spezifisch für das Experiment implementiert und es als Teil eines *Open-Source* Daten-Frameworks mit dem Namen *Neo* veröffentlicht. *Neo* bietet den Vorteil Daten in Form einer standardisierten, internen Struktur bereitzustellen, die es Wissenschaftlern ermöglicht gemeinhin gültige Analyseroutinen für unterschiedliche Datenformate zu implementieren. Darüber hinaus ist es möglich die *Neo* Strukturen mit experimentspezifischen Informationen der Daten zu annotieren. Um solche Informationen, die man auch als Metadaten bezeichnet, automatisch integrieren zu können, sollte man diese am Besten in einem maschinenlesbaren Format organisieren. Obwohl verschiedene Softwarelösungen für solche Metadatenformate existieren, wurden diese in der Regel nicht für komplexe Anwendungsfälle wie dem Beispielexperiment getestet bzw. entlang eines solchen entwickelt. In den meisten Fällen bieten sie nur ein Framework für eine standardisierte Metadatendarstellung oder -spezifikation an, lassen jedoch einen Lösungsansatz für die Kompilierung einer Metadatensammlung vermissen. Für das Beispielexperiment habe ich mich für das Metadatenframework mit dem Namen *odML* (*open metadata Markup Language*) entschieden, ein *Open-Source* Projekt, welches von dem deutschen Ableger (*German-Node / G-Node*) der Internationalen Neuroinformatik Koordinationseinrichtung (*International Neuroinformatics Coordination Facility / INCF*) entwickelt wurde. In meiner Doktorarbeit werde ich demonstrieren wie Metadaten für das

Beispielexperiment organisiert werden können, um andere Wissenschaftler in die Lage zu versetzen, eine entsprechende *odML* Metadatensammlung zu erstellen und zu nutzen. Um die Erstellung für meine Kooperationspartner zu erleichtern, habe ich ein *Python*-Softwarepaket mit dem Namen *odMLtables* entwickelt, welches den Zugriff auf das *odML* Framework vereinfacht, indem es eine Transformation von *odML* in ein gängiges Tabellenformat (csv oder xls) und wieder zurück zu *odML* ermöglicht. Außerdem habe ich einen vollständigen Arbeitsablauf zur Zusammenstellung und Speicherung von Metadaten des Beispielexperiments in eine umfassende *odML* Dateisammlung erstellt. Zusätzlich habe ich eine spezifische Datenladeroutine zur Verfügung gestellt, die die *Neo* Datenstrukturen mit entsprechenden Metadaten annotiert. Diese Erweiterung verbesserte den Arbeitsablauf einer neurowissenschaftlichen Analyse der Daten des Beispielexperiments, was ich mit dem letzten Teil meiner Arbeit demonstriere. Zusammengefasst zeige ich, dass die Vorbereitungen um Forschungsdaten fachgerecht innerhalb von wissenschaftlichen Kooperationen zu teilen zwar mühselig und langwierig, aber dennoch essentiell sind um erfolgreich Daten und deren Analyseergebnisse für eine breite Zielgruppe zu veröffentlichen. Um die gemeinsame Nutzung von Daten innerhalb der neurowissenschaftlichen Gemeinde zu fördern und um ein besseres Fundament für reproduzierbare Forschung zu schaffen, biete ich mit meiner Doktorarbeit eine ausgewählte Selektion von verfügbaren Technologien eine kohärente Strategie elektrophysiologische Daten und Metadaten zu verwalten.

Contents

1 Introduction **15**

1.1 Research - to share or not to share? . . . 16
1.2 Metadata frameworks - trivial concept, complex implementation . . . 21
1.3 Data frameworks - making data access easy . . . 25
1.4 Putting the concepts to the test... (thesis overview) . . . 31

2 Descriptor of the example experiment **33**

2.1 The task(s) . . . 36
2.1.1 Two-instructions paradigm (standard task) . . . 36
2.1.2 Other tasks . . . 38
2.2 The subjects . . . 40
2.2.1 The training and general performance . . . 40
2.2.2 The array: implant, recording quality, removal and location . . . 42
2.3 Setup . . . 43
2.3.1 Experimental apparatus . . . 46
2.3.2 Behavioural control system . . . 49
2.3.3 Neuronal recording platform . . . 53
2.4 Technical validations . . . 56
2.4.1 Digital event translation . . . 58
2.4.2 Behavioural event extraction . . . 58
2.4.3 Offline spike sorting . . . 59
2.4.4 Quality assessment . . . 62
2.4.4.1 Correction of data alignment . . . 62
2.4.4.2 Data gaps . . . 62
2.4.4.3 LFP data . . . 64
2.4.4.4 Spike data . . . 65
2.5 Metadata and data accessibility . . . 66
2.5.1 Machine-readable metadata - organization matters . . . 67
2.5.2 Data access . . . 72
2.5.2.1 Task 1: providing a generic BlackrockIO for Neo . . . 73
2.5.2.2 Task 2: annotate Neo data objects with more metadata . . . 74
2.6 What is going to be published in *Scientific Data* . . . 75

3 Metadata management **79**

3.1 Why make the effort to create a comprehensive metadata collection? . . . 80
3.2 Management strategies for a comprehensive metadata collection . . . 84

3.2.1 Distribution of information . . . 86
3.2.2 Structuring information . . . 87
3.2.3 Workflow . . . 89
3.3 Using an odML metadata collection . . . 94
3.3.1 Manual inspection . . . 94
3.3.2 Navigating the odML structure . . . 97
3.3.3 Navigating across odML-files . . . 102
3.3.4 Integration of additional metadata . . . 102
3.3.5 odML access via Matlab . . . 105

4 Publishing research of the example experiment 109
4.1 Background of the scientific question . . . 110
4.2 Material and Methods . . . 111
4.2.1 Power spectra and coherence . . . 113
4.2.2 Definition of vector fields . . . 114
4.2.3 Quantification of observed phase patterns . . . 115
4.2.4 Classification of spatial phase patterns . . . 117
4.3 Results . . . 119
4.3.1 Spectral LFP properties . . . 120
4.3.2 Identification of phase patterns . . . 120
4.3.3 Relation of power and phase patterns to behaviour . . . 127
4.3.4 Quantification of phase patterns . . . 128
4.3.5 Beta amplitude determines phase pattern . . . 132
4.4 Discussion of the research results . . . 132

5 Conclusion 137
5.1 Outlook - Paving the way for new scientific findings . . . 145
5.2 Outlook - Aiming towards reproducible neuroscience . . . 149

Bibliography 157

List of Figures

1.1 Influencing factors when documenting an experiment to share data across different levels 17
1.2 The odML metadata framework: overview of the structural design 26
1.3 The Neo data framework: overview of the Neo objects and their relations 29

2.1 Schema of the standard reach-to-grasp task 37
2.2 Overview of array implants in left brain hemispheres 44
2.3 Overview of array implants in right brain hemispheres 45
2.4 Overview of the setup of the reach-to-grasp experiment 47
2.5 Overview of the target object system 48
2.6 Overview of the behavioural control system 51
2.7 Overview of the neuronal recording platform 54
2.8 Overview of the electrode configuration of the implanted Utah-Arrays 55
2.9 Overview of sortable single unit activities across the arrays 61
2.10 Metadata sources of the reach-to-gasp experiment. 68

3.1 Emergence of metadata sources over time in the R2G experiment 81
3.2 Example of a formalized metadata base data selection 81
3.3 Example of a metadata analysis 82
3.4 Example Python code to create an odML-file 85
3.5 Schematic view of the odML structure for metadata of a Utah-Array 88
3.6 Schematic workflow for generating odML-files in the R2G experiment. 90
3.7 Example of an odML-XLS template 92
3.8 Workflow of odMLtables 92
3.9 XML representation of an odML-file 95
3.10 Editor view of an odML-file 95
3.11 Browser view of an odML-file 96
3.12 Extract unique objects from an odML file in Python 98
3.13 Extract ambiguous objects from an odML-file via search conditions in Python 99
3.14 Extract ambiguous objects from an odML-file via partial searches in Python 100
3.15 Extract all ambiguous objects from an odML-file in Python 101
3.16 Extract unique objects from a set of odML-files for a data selection in Python 103
3.17 Extract multiple objects from an odML-file for a data selection in Python 104
3.18 Load an odML-file in MATLAB - “tree” option 106
3.19 Load an odML-file in MATLAB - “odml” option 106
3.20 Extract unique objects from an odML file in MATLAB 106
3.21 Extract ambiguous objects from an odML file in MATLAB 107

3.22 Extract all ambiguous objects from an odML file in MATLAB 107

4.1 Code examples illustrating the combined data and metadata access 112
4.2 Explanation of the methodology 116
4.3 Selection of thresholds used for automatic phase pattern classification 118
4.4 Spectral properties of the LFP 121
4.5 Extraction of phase and amplitude maps 122
4.6 Phase patterns and their detection 124
4.7 Behavioural correlates and relation to average beta power 126
4.8 Statistics of detected phase patterns 129
4.9 Correlation of instantaneous beta power and spatial pattern of phases 131

5.1 Survey on workflows in electrophysiology - selected questions 138
5.2 Survey on metadata acquisition and handling in neuroscience - selected questions . . . 151
A.1 Overview of data types contained in i140703-001 170
A.2 Overview of data types contained in i140703-001 171
A.3 Overview of identified single and multi unit activities of l101210-001 and i140703-001 . 173
A.4 Overview of LFP and spike data of l101210-001 and i140703-001 173
A.5 Data selection for preliminary study based on task and trial numbers 175
A.6 Data selection for preliminary study based on averaged trial cycle 175
A.7 Data selection for preliminary study based on signal quality 177
A.8 Data selection for preliminary study based on reaction times 177
A.9 Beta profile peak in preparatory delay period is shifted with increasing RT 178
A.10 Beta profile differences in preparatory delay period for grip and force types 179
A.11 Example of trial-by-trial variability of beta profile analysis 180

List of Tables

2.1 Reach-to-grasp monkeys: dates and facts . . . 35
2.2 Possible conditions of the standard reach-to-grasp task . . . 38
2.3 Overview of the attached object sensors monitoring the monkey's grasp behaviour . . 49
2.4 Overview of digital trial events . . . 57
2.5 Overview of trial performance codes . . . 58
2.6 Overview of data gaps caused by a package loss in the ns5-files . . . 63

4.1 Datasets used in the presented research publication . . . 111
4.2 Overview of threshold conditions used for phase patterns . . . 119
4.3 Occurrence of pattern classes . . . 125

A.1 Overview of recording days of the datasets that will be published . . . 169
A.2 Overview of datasets that will be published . . . 169
A.3 Overview of trials performed during the published datasets . . . 172
A.4 Overview of offline sorted single and multi unit activity . . . 172
A.5 Overview of selected trials for the trial-averaged beta profile analysis . . . 176
A.6 Overview of offline sorted datasets of the first recording period of monkey T . . . 181
A.7 Overview of offline sorted datasets of the second recording period of monkey T . . . 182
A.8 Overview of offline sorted datasets of monkey L (year 2011) . . . 182
A.9 Overview of offline sorted datasets of monkey L (year 2010) . . . 183
A.10 Overview of offline sorted datasets of the first recording period of monkey N . . . 184
A.11 Overview of offline sorted datasets of the second recording period of monkey N . . . 184

Chapter 1

Introduction

During the last decades, we have witnessed an increasing awareness for the importance of ensuring that the results of scientific endeavours are based on research which can be, first of all, re-analysed, and, second of all, re-used. This development is not only provoked by cases of scientific misconduct (Pulverer, 2015b), but mostly stimulated by the challenges of increasingly complex and technologically advanced experimental research projects which need to be managed in larger interdisciplinary teams. In addition, the complexity of the underlying experiments and their data hold the potential to address multiple scientific questions often exceeding the expertise of the original community conducting these studies and demanding revisions by third parties (Open Science Collaboration, 2015; Fienberg et al., 1985). These developments are reflected in the increasing number of journals asking for the underlying data and the exact analysis workflow of the research articles (Candela et al., 2015), the increasing number of larger research collaborations to solve demanding scientific questions (Adams, 2012), and the increasing number of funding agencies linking their support for a research project to better accessibility of the data and the analysis results for the scientific and public community (reported in Borgman, 2012).

Sharing data in interdisciplinary collaborations or with unknown third parties requires, however, a more structured and standardized mode of working than in the past, where scientific workflows were largely modelled after personal preferences and the reproducibility of the analysis results was relying on the knowledge space of often only one researcher. This raises the question of what are the prerequisites to accomplish such a modern scientific workflow and how it can be mastered especially in neuroscience, a highly interdisciplinary scientific field?

In this thesis, this question will be addressed by means of an electrophysiological experiment with behaving monkeys originating from a interdisciplinary collaboration between the experimental Cognitive Motor Control (CoMCo) group at the INT (Institut de Neurosciences de la Timone), of the Aix-Marseille University in France and the theoretical Statistical Neuroscience (SN) group at the INM-6 (Institute for Neuroscience and Medicine 6, Computational and Systems Neuroscience) of the Jülich Research Centre in Germany. Although this example experiment is a specific case scenario, I derive general guidelines for the management of electrophysiological data and metadata which will help other researchers to maintain, for themselves and for others, accessibility to their experiments.

Before giving an overview on the content of the main chapters of this thesis, I will first introduce the concepts of data sharing (Section 1.1), metadata (Section 1.2) and data frameworks (Section 1.3).

1.1 Research - to share or not to share?

The penultimate paragraph of the following section is in parts based on an extract of the introduction of the manuscript "Handling Metadata in a Neurophysiology Laboratory", which was published in Frontiers of Neuroinformatics (Zehl et al., 2016).

Publicly releasing research data for use by others started to become a topic of discussion three decades ago (Fienberg et al., 1985), and the scientific community almost universally agrees that sharing data can only be beneficial. As a result, nowadays, most funding agencies require a management plan that ensures the preservation and accessibility of data with each incoming grant proposal, and more and more peer-reviewed journals at least encourage the deposit of data in public archives (Borgman, 2012; Pampel and Dallmeier-Tiessen, 2014). Actually, for research data the question *"to share or not to share?"* should not even arise, because sharing data is an unavoidable part of any research project and happens to different extent on different levels:

Level A) Share your data with yourself.

Level B) Share your data with your laboratory.

Level C) Share your data within a collaboration with external partners.

Level D) Share your data with unknown third parties.

On all of these levels data sharing does not just mean physically providing the data, but also making their context, meaning, problems and scientific potential accessible in a proper documentation. This also includes the provision of tools to load and work with the data. If one goes along the data sharing levels from A to D, one can realize two factors influencing the difficulty of actual data sharing (cf. Figure 1.1): (i) the number of involved researchers is increasing, and (ii) the shared knowledge space among the involved researchers is decreasing. In the following, I would like to explore how these two factors influence the documentation of the data and the provision of corresponding software tools to access and work with the data and why sharing data across the different levels is not a trivial task.

Before sharing the data with others, an individual researcher must consider sharing the data with her/his future self (level A). It is evident that our individual knowledge space changes over time. In particular, specific knowledge about an experiment may not be memorized after some time. For this reason, researchers should treat themselves as strangers to guarantee the full comprehension of the experiment for later in the day, for tomorrow, next week, month or year. However, it is not an easy task to reflect on ones own actions all the time and keep an outward perspective on the documentation and tools to access the data (Zehl et al., 2016). For this reason, sharing data even with oneself entails pitfalls which are exemplified in the testimonies stated in Figure 1.1 (in blue triangle).

These stated situations may seem to be caused by a disorganized researcher and, indeed, with increasing personal organization such issues can be avoided to a certain degree. Nevertheless, most researchers will recognize these or similar problems regarding their past or current projects, because how to accomplish a reliable self-organisation to fully avoid these pit falls, is usually not taught during the course of studies, but grind out over time and experience. Furthermore, the personal knowledge space of a researcher changes with experience which can lead to new perceptions on a running experiment increasing the importance to document aspects of formerly minor interest. To plan these unforeseeable developments in advance, at least to a certain degree, is a very difficult task. In addition, the complexity of today's experiments aggravates to transfer experience and solutions gained in individual projects. As a result, documentation of an experiment and tools to access the data

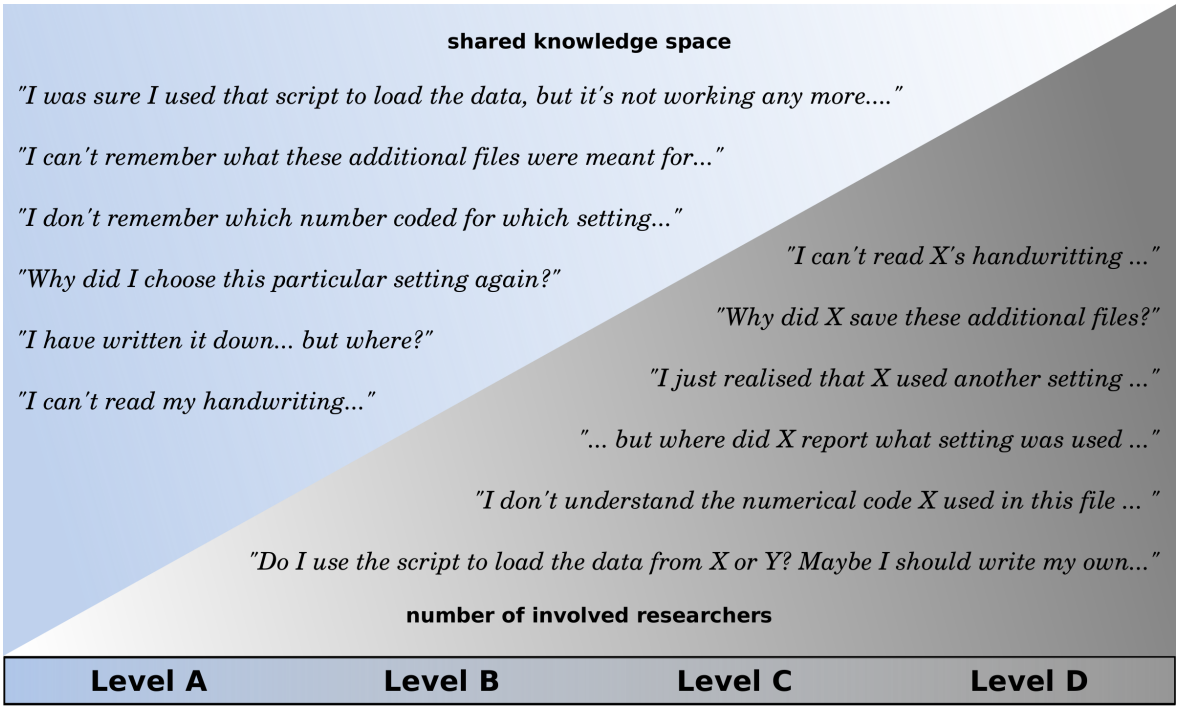

Figure 1.1 – Influencing factors when documenting an experiment to share data across different levels. Within a research project one usually shares data with oneself (Level A), with members of the same laboratory (Level B), with external collaboration partners (Level C), and with unknown third parties (Level D). There are two major factors that influence the difficulty to share data across these levels: increase of the number of involved researchers (indicated by grey coloured triangle) and the decrease of shared knowledge space among these involved researchers (indicated by the blue coloured triangle). Assuming an increasing time line of a research project (bottom square) one can state that sharing data happens on Level A immediately, followed by Level B and C and usually towards the end of the project on Level D. Considering this, the shared knowledge space decreases over time even for an individual researcher (as indicated by the testimonies stated in the blue triangle) and with the increasing number of involved researcher which aggravates the communication about the experiment (as indicated by the testimonies stated in the grey triangle).

are usually project specific and limited to the sight and knowledge space of an individual researcher, making it difficult for others to extract the necessary information to understand and use the data of the experiment.

This leads to problems in level B of data sharing, because nowadays most scientific projects are indeed not conducted by one individual researcher, but by a group of researchers within a research laboratory. Such a group typically consists of at least two persons, the leading principal investigator (PI) and a student. Some complex projects may even involve several students, technical assistants, and PIs from the beginning to cope with the amount and complexity of the data. If on level A of data sharing, one only had to consider the time-developing knowledge space of one individual person, one now has not only to consider this change of knowledge space for all members of the group, but also that the individual knowledge spaces of the group members not necessarily fully overlap. Even if researchers are from the same or a similar disciplinary field, as it is often the case for members of one laboratory, their individual knowledge about the experiment is separated by the differences in their education, their experience and their daily routines. The success of producing and sharing data within a laboratory is therefore influenced by the quality of communication between the involved personnel. One can easily adapt the previous testimonies to situations where the communication between researchers failed (cf. Figure 1.1, in grey triangle).

Within a laboratory, these miscommunications can be certainly solved in one-on-one interviews, but this approach will fail as soon as one of the involved researchers leaves the project and is not directly reachable any more. A change of personnel is not uncommon (e.g., the exchange of students), especially for long-term projects where the experiment is running over years. As a result, the documentation of an experiment and tools to access the data within a laboratory are usually shaped by one researcher (e.g., the PI), but then extended and adapted over time by all further group members. To guarantee the accessibility of such an experiment over time, the involved researchers need to agree on a common strategy which can be complied by the current group members and easily taught to new personnel. To develop such a strategy costs time and effort, and is therefore usually limited to the specific needs and goals of the laboratory. Furthermore, these often home-brewed solutions are usually not built on a solid software foundation and for this reason error-prone. Such non-established solutions make it again difficult for outside scientists to extract the necessary information to understand the experiment and its data.

This brings us to level C of data sharing. Nowadays the analytical exploitation of complex datasets often calls for larger interdisciplinary collaboration teams (Greene, 2007; Leydesdorff and Wagner, 2008; Whitfield, 2008). Besides a pure increase in further personnel, communication between members of such collaborations is often not only negatively influenced by the geographical, but also by the interdisciplinary distance. External collaboration partners of an experimental laboratory need to understand the context, meaning, and problems of the provided data to be able to perform meaningful analysis and avoid misinterpretation of the corresponding results. This understanding is necessary independent of the quality and nature of the provided data which could include the complete original raw dataset, or only derived data parts from first pre-processing steps (e.g., spike trains after spike sorting) which are then forwarded to the external partners for further analysis. The group conducting the experiment must therefore consider in the documentation of the data that the shared knowledge space with the collaboration partners is rather small. Furthermore, to create the most useful documentation for an experiment, it should be not only human, but also machine readable. Especially in the absence of appropriate tools, the latter requires usually advanced programming knowledge which experimental researchers often still lack. Disciplinary preferences of tools and programming languages used to access and work with the experimental data additionally complicate the collaborative work.

If such disciplinary preferences cannot be overcome, it is helpful to at least base the documentation of the data on a format which is generic enough to be used or to be made usable in any tool and programming language. Hence, the success of collaborations relies on the development of a strategy to document and access the data which is accepted and validated across the involved research labs. Often, ad-hoc strategies are put in place, but they often do not pay off in the long term. Finding instead an established agreement between labs can be cumbersome and time consuming, but is necessary nevertheless.

Despite all difficulties in how to best document and how to best provide access to the data of an experiment, the inevitability of sharing data and the associated immediate benefits (e.g., manpower and broader expertises) led to an increase of research collaborations within or across laboratories, where a larger group of interdisciplinary scientists is working on data of the same experiment (Greene, 2007; Leydesdorff and Wagner, 2008; Whitfield, 2008; Adams, 2012). Unfortunately, it is still not commonly practised to publish data for the use of third parties (level D), although most peer-reviewed journals require, besides a full description of the new scientific findings and used methods, at least also a brief summary of the used data and encourage their deposit in a public repository (e.g., Nature (2005) for Nature Publishing Group, Hanson et al. (2011) for Science, Bloom et al. (2014) for PLoS journals, and Alsheikh-Ali et al. (2011) as review on data policies in high-impact journals). Instead, to this day, a high percentage of publications do not provide the material needed for the verification of the research findings and possible re-use of corresponding data by other researchers - not even when requested (Campbell et al., 2002; Wicherts et al., 2006; Savage and Vickers, 2009; Open Science Collaboration, 2015; Baker, 2015). For a complete open scientific inquiry, to promote best scientific practice, and to provide every opportunity to verify research findings, it would, however, be best to provide the actually used data along with a proper documentation (Borgman, 2012; Pulverer, 2014; Pampel and Dallmeier-Tiessen, 2014). This can be realised by either publishing research data as an independent information object in a data repository, as integral or supplementary part of an original research article (enriched publication) or as stand-alone data publication (Dallmeier-Tiessen et al., 2012; Pampel and Dallmeier-Tiessen, 2014). Especially the latter has the potential to promote secondary analysis, and extend the data access to researchers of third parties with perspectives and expertises outside of the original addressed community which can trigger new developments of theories, techniques, or scientific findings (Fienberg et al., 1985; Candela et al., 2015). Additionally, data publications can be very useful as resources for training scientific techniques or testing new scientific methods (Fienberg et al., 1985; Tenopir et al., 2011). Sharing data with third parties, thus, holds essential benefits for scientific progress. For this reason, most funding agencies demand a data management plan with each incoming grant proposal that ensures the preservation and accessibility of the data created in the funded project (e.g., National Institutes of Health, 2003[1];Foundation, 2014[2];Doldirina et al., 2015[3]). With the overall benefits for science, the encouragements by journals, and the demands by funding agencies, why do scientists still generally hesitate when it comes to share data with third parties (level D)? The reasons for this ongoing hesitation are mainly based on tow barriers in science (Fienberg et al., 1985; Paton, 2008; Tenopir et al., 2011; Borgman, 2012; Pampel and Dallmeier-Tiessen, 2014; Candela et al., 2015), which I will explain in the following paragraphs.

The difficulties of providing a proper, understandable documentation of an experiment and guaranteeing easy access to the corresponding data are often already misjudged in a closed group of scientists

[1] https://grants.nih.gov/grants/policy/data_sharing/data_sharing_guidance.htm

[2] http://www.nsf.gov/bfa/dias/policy/dmp.jsp

[3] https://ec.europa.eu/jrc/en/publication/eur-scientific-and-technical-research-reports/jrc-data-policy

(level A, B, and C). The reason for this misjudgement is based on the fact that, at least in the beginning of a research project, a personal communication between researchers is often possible. However, the assumption to be able to personally communicate with each other if questions or problems arise, leads in most cases to blanks in the documentation. Developing a strategy to document and access the data which is accepted and validated across the involved researchers is certainly difficult and time consuming and the first barrier which needs to be overcome. The pressure of funding agencies on scientists to produce results mainly in form of new publications (Editorial, 2005; Neill, 2008; Fanelli, 2010), unfortunately contributes to the behaviour that experimental data are primarily produced to achieve a research goal and not because they are going to be shared with unknown third parties (Fienberg et al., 1985). As a consequence, the degree of documentation and quality of implemented tools to access the data are often reduced to the minimum amount necessary for only the involved researchers and a specific research question. Moreover, if the time course of planning and conducting an experiments is rather long, the willingness to share data with third parties is further decreased, because of the fear that subsequent researchers may publish results before the original researcher or research group (Borgman, 2010; Klump, 2012). If further the elicitation of the data involved a huge amount of time and effort, the fear of the original researcher or research group to loose control over their experimental data and their original purpose is naturally provoked (Fienberg et al., 1985). If sharing data with third parties, although it generally serves scientific progress, has, thus, no immediate benefits for the researchers and may even negatively influence their publication process, the question *"Why should we make the effort?"* is completely understandable. Providing credit through formal citations of data publications is just a first step to overcome this barrier.

The second barrier is the sheer complexity of nowadays experiments and the fact that generic software solutions for managing and documenting corresponding data are so far missing. Especially for electrophysiological experiments the costs of managing and documenting data, and, with that, of sharing them, are extraordinarily high. The reason for this lies in the complexity of the experiments, caused already by the heterogeneous nature of neuronal signals and the associated historic development of an equally heterogeneous amount of hard and software tools and formats as well as various experimental approaches for collecting, storing and analysing the corresponding data (Denker and Grün, 2015). Just during the last decades, the technological progress in neuroscience has led to methods that enable the simultaneous recording of activity from tens to hundreds of neurons, *in vitro* or *in vivo*, using a variety of techniques (Obien et al., 2014; Verkhratsky et al., 2006; Nicolelis and Ribeiro, 2002) in combination with sophisticated stimulation methods, such as optogenetics (Deisseroth and Schnitzer, 2013; Miyamoto and Murayama, 2015). In addition, recordings can be performed in parallel from multiple brain areas, together with behavioural measures such as eye or limb movements (Vargas-Irwin et al., 2010; Maldonado et al., 2008). Such recordings enable scientists to study network interactions and cross-area coupling and can be used to relate neuronal processing to the behavioural performance of the subject (Lewis et al., 2015; Lisman, 2015; Berenyi et al., 2013). These approaches led to increasingly complex experimental designs that are difficult to parametrize, e.g., due to multidimensional characterization of natural stimuli Geisler (2008) or high dimensional movement parameters for almost freely behaving subjects Schwarz et al. (2014). It is a serious challenge for researchers to keep track of the overwhelming amount of documentation needed at each experimental step and to precisely extract all information relevant for data analysis and the interpretation of results. Various aspects need to be considered, such as the parametrization of the experimental task, broken hardware (e.g., electrodes), condition of the subject, filter settings, sampling rates and quality of the recorded data, or preprocessing steps necessary before analysing the data (e.g., spike sorting). Nevertheless, the organization of such information, termed metadata, is of utmost importance for conducting research

in a reproducible manner, i.e., the ability to faithfully reproduce the experimental procedures and subsequent analysis steps (Laine, 2007; Tomasello and Call, 2011; Peng, 2011). Moreover, detailed knowledge of the complete recording and analysis processes is crucial for the correct interpretation of results, and is a minimal requirement to enable researchers to verify published results and build their own research on the previous findings. Nevertheless, the heterogeneity makes neuroscientific and especially electrophysiological experiments particularly complex to understand and with this particularly difficult to document (Zehl et al., 2016).

Generic software solutions for the documentation of such complicated experiments as well as the access of the corresponding data are urgently needed to simplify and promote data sharing in the neuroscientific community. Disciplines which are characterized by a more homogeneous development of standards and techniques or which were forced to create them earlier in order to keep progressing (e.g., social science, climate research, genetics, proteomics, etc.), have already developed a more open and successful culture of sharing data with third parties (Tenopir et al., 2011; Borgman, 2012). In contrast, due to the heterogeneity of the neuroscientific field, available generic software solutions are unfortunately still not easily accessible for neuroscientific experiments and their implementation into the daily routines of an individual research project involves a lot of time and effort. In Section 1.2 and Section 1.3 I will further introduce what documenting electrophysiological experiments comprises and which metadata and data framework can be used to simplify data access within or outside of research collaborations in the field of neuroscience.

1.2 Metadata frameworks - trivial concept, complex implementation

The following section is mainly based on text parts of the manuscript "Handling Metadata in a Neurophysiology Laboratory", which was published in Frontiers of Neuroinformatics (Zehl et al., 2016).

As elaborated above, a proper and thorough documentation of an experiment is crucial for sharing data, but its compilation is a time consuming process. The most complete and comprehensible way to document an experiment is (i) to provide a full description of the background story, the details, the workflow, and the peculiarities of an experiment and its data in form of a continuous text (e.g., for a data publication, cf. Chapter 2), and (ii) to comprise all this knowledge in a more standardized fashion as part of a metadata software framework (cf. Chapter 3). With (i) the responsible researcher can provide a human readable (personalized) documentation of the experiment, while with (ii) she/he is forced to standardize her/his knowledge and making it additionally machine-readable. Metadata that are not only human, but also machine-readable, can be used replicably during automatized analysis processes. The realization of both documentation modes (i and ii) are needed to make a step forward towards reproducible research workflows. To explain why in particular the compilation of (ii) is not a straight forward process and, for this reason, is still often incomplete (or missing) for contemporary neuroscientific experiments, I will first introduce again the term "metadata".

Metadata is generally defined as data describing data (Baca, 2008; Webster, 2006). More specifically in the context of this thesis, metadata are information that describe the conditions under which a certain dataset has been recorded (Grewe et al., 2011) and how the data were preprocessed before they are usable in any subsequent analysis. For this, ideally all metadata should be available in a single machine-readable format that can be linked to the corresponding datasets of an experiment. In practice, however, this situation is, particularly for electrophysiological experiments, difficult to

achieve. Already the fact that an electrophysiological setup is composed of several hardware and software components, often from different vendors, imposes the need to handle metadata distributed over multiple files of different formats. Some files may even contain metadata that are not immediately machine interpretable, meaning they cannot be directly accessible for technical applications or algorithms. Furthermore, especially *in vivo* experiments require the full attention of the experimenters, which limits the amount of time to consider how to capture metadata thereby incurred. For this reason, metadata that arise unexpectedly during an experiment, e.g., the cause of a sudden noise artefact, are commonly documented as quick handwritten notes in a laboratory notebook. In fact, handwritten notes are often unavoidable, because legal regulations of some countries, e.g. France (see www.cnrs.fr/infoslabos/cahier-laboratoire/docs/cahierlabo.pdf), require the documentation of experiments in the form of handwritten laboratory notebooks.

Putting the problem of documentation into the context of collaboration, the documentation problems gains another dimension. To perform and document experiments and share the corresponding metadata and data with colleagues, experimenters typically develop their own lab procedures and practices (cf. Section 1.1, level A and B of data sharing). Within the same laboratory, these procedures often include the transmission of metadata by personal communication, through handwritten notes or implicitly by commonly trained experimental procedures (cf. Section 1.1, level B of data sharing; shared knowledge space). However, at latest when it comes to data sharing across laboratories such practices are doomed to fail (cf. Section 1.1, level C of data sharing). Especially in interdisciplinary collaborations (e.g., an experimental and theoretical group), implicit domain-specific knowledge is often not communicated or is communicated in an ambiguous fashion that leads to misunderstandings. Furthermore, if metadata are distributed over several unstandardised formats (e.g., handwritten notes), essential information is often missed in the exchange (Open Science Collaboration, 2015; Hines et al., 2014). For this reason, "*just*" having metadata somewhere documented is unfortunately only half the battle for a proper documentation. The general principle should be to organize and comprise metadata into a comprehensive collection of a machine-readable, standardized format which can be linked to the corresponding datasets they describe. The possibility to annotate data with metadata in a clear and concise fashion is helpful to avoid miscommunication in how to use the data in a collaborative work (cf. Section 1.1, level B and C of data sharing), but also an important step to perform and publish replicable analysis (cf. Section 1.1, level D of data sharing).

Organizing the content for such a metadata collection as well as compiling it are, however, time-consuming processes that need manpower, especially if such a metadata management is not planned ahead. Although several software solutions for such metadata frameworks exist, they are usually not tested for complex experiments and in most cases do not provide guidance for actually compiling a useful metadata collection. In fact, setting up a corresponding workflow requires usually higher programming skills, especially when the compilation of the metadata collection and the integration of the metadata framework into the analysis workflow should be automatized. Which software solution to choose as metadata framework highly depends on individual preferences. Nevertheless, from my own experience as member of an interdisciplinary research collaboration team working on data from a complex behavioural experiment that involves electrophysiological recordings from a large number of electrodes from several subjects over a period of years, I can summarize the following requirements (reqt) for a useful metadata framework:

reqt 1) The metadata framework should represent metadata in a human and machine-readable format. As mentioned above the metadata collection should comprise the continuous text version of the documentation of the experiment and its data in a more standardized format.

To gain quick overviews of a specific recording, e.g., the used hardware settings, it can be quite useful for researchers to be able to manually inspect the compiled metadata collection. It is therefore highly recommendable to choose a metadata framework that offers software tools providing convenient human-readable access to metadata. On the other hand, it sometimes can be helpful to gain an overview of the metadata of an entire experiment. To be effective it is important that such a screening process can be automatized or is supported by software tools that allow the scientist to extract metadata interactively, which addressed specific research questions. For this, the format of the used metadata framework should be necessarily machine-readable.

reqt 2) The metadata framework should define metadata in standardized key-value pairs. Metadata can describe various aspects of the recording and preprocessing steps of the experiment. Nevertheless, all metadata can be represented by a key-value pair, where a semantic key names the metadata, and the value contains the metadata item. It is also helpful if the key-value pairs can additionally be annotated with a more elaborated definition of what they represent. Commonly defined key-value pairs are very helpful to convey relevant metadata in a precise way, even in the presence of communication barriers between researchers (e.g, resulting from infrequent and impersonal communication). If possible, this semantic metadata definition can be even more generalized by linking to commonly defined ontologies (Uschold and Gruninger, 1996; Bowden et al., 2007; Larson, 2009; Guarino et al., 2009).

reqt 3) The metadata framework should be flexible in content. Metadata can be of arbitrary type and could even be not extractable from their original source (e.g., an image). For this reason it is important that the metadata framework can handle all types of extractable metadata, and, in addition, can point to a file or a remote location of a metadata source, if the source cannot be extracted.

reqt 4) The metadata framework should be flexible in structure. As elaborated before, especially neuroscientific experiments are complex and heterogeneous, but it is also in general safe to say that each experiment has its own peculiarities. In neuroscience usually each laboratory has its own way of planning and conducting their experiments using their very own compilation of hard and software components, either from vendors or self-made. It is therefore of utmost importance that the organization of metadata in the framework is highly flexible and adjustable to the needs and conventions of the individual laboratory. Providing commonly accessible standards (cf. reqt 1 and 2) without withdrawing the individual scope for design from the scientists, is one of the key arguments to decide for or against the usage of a specific metadata framework.

reqt 5) The metadata framework should provide the possibility to enrich the metadata collection over time. A difficult aspect of metadata management is that one cannot know all metadata in advance that are necessary to ensure the reproducibility of the experiment and the data analysis. Aggregation of additional metadata over time is usually unavoidable, and thus the metadata collection needs to be enriched when new information becomes available. An example for such an enrichment of a metadata collection is the integration of results and parameters of preprocessing steps applied to the recorded data, since these are usually performed long after the primary data acquisition (e.g., spike sorting).

reqt 6) The metadata framework should be accessible via different programming languages. In research collaborations it is not unusual that the partners base their work on

different software technologies and use different programming languages. To guarantee access to the metadata collection across such a collaboration, it is therefore necessary that the chosen metadata framework provides various application program interfaces (APIs) for different programming languages (e.g., Matlab and Python).

reqt 7) The structure of the metadata framework must be searchable. Depending on the experiment, the structure of a metadata collection can become large and complex which makes it difficult to find certain metadata. For this reason the metadata framework should provide functionality which can be used to find and extract metadata with minimal user knowledge on the actual organization of the metadata collection. On the one hand, this requirement supports the human-readability of the metadata framework (reqt 1), on the other hand, the compliance of the machine-readability of the framework (reqt 1) using standard key-value representations of the metadata (reqt 2) is crucial to address/fulfil this requirement.

reqt 8) The metadata framework should provide the opportunity to be linked to corresponding data. Data of an experiment can usually be used to address multiple scientific questions. In this context, it is necessary to select data according to certain criteria which are defined by the requirements and constraints of the scientific question. These criteria are usually represented by metadata specific for the corresponding data. To automatize the data selection, it is essential that the metadata of the used framework can be linked to the data they describe. Such metadata annotations not only facilitate the data selection procedure, but also make it more robust and with this more reproducible. Moreover, in combination with the usage of semantic metadata definitions (cf. reqt 2) corresponding annotated datasets could be published following the concept of linked-data which would allow scientists to connect and query data from different sources, therefore enable researchers to share data across laboratories and with third parties more easily and efficiently (Halb et al., 2008; Glasson and Hussain, 2008; Bizer et al., 2009; Freitas et al., 2012; Aoki-Kinoshita et al., 2015; Zhou et al., 2016).

For the example experiment (see Chapter 2), I decided to implement the comprehensive metadata collection using the open metadata Markup Language (odML), a metadata framework solution that nearly fulfilled all the above mentioned functionalities. I chose the odML framework especially because it is comparatively generic and flexible, which makes it well-suited for a broad variety of experimental scenarios, and which enabled the laboratory to introduce it as metadata framework in multiple running collaborations.

The odML framework was introduced by Grewe et al. (2011) as a simple file format in analogy to SBML in systems biology Hucka et al. (2003), or NeuroML in neuroscientific simulation studies (Crook et al., 2012; Gleeson et al., 2010). The odML format is based on XML, which is a textual data format. It is therefore immediately possible to read and (manually) edit an odML-file with any available text editor, thus human-readable (reqt 1). For larger metadata structures the XML representation of an odML-file is, though, inconvenient to read. For this reason, the odML framework provides (i) an XSL stylesheet that makes it possible to view odML-files via a common web browser, and (ii) a general user interface (GUI) in form of an editor. In all three possibilities to manually inspect an odML-file, the underlying hierarchical structure of the metadata framework is evident. The odML metadata framework is built on the concepts of Properties (keys) paired with Values as the structural foundation of a metadata file (reqt 2). A Property may contain one or several Values. Its name is a short identifier by which the metadata can be addressed and its definition can be used to give a textual description of metadata stored in the Value(s). To store metadata of various types or to enter a link to a file or directory to more complex metadata, the odML framework internally defines a number

of data types (including URLs) for the Value(s) (reqt 3). Additionally, the Value(s) can be further defined by the possibility to set a unit and uncertainty. In the odML framework, Property-Value(s) pairs that belong to the same context (e.g., description of an experimental subject) are then grouped in so called Sections which are specified by their name, type and definition. Sections can further be nested, i.e. contain sub-sections and thus form a tree-like structure. At the root of the tree is the Document which can contain information about the author, the date of the file and a version.

Figure 1.2 illustrates how a subset of experimental metadata can be organized in an odML-file. Although the odML framework provides terminology blueprints[4] for common electrophysiological metadata, the content and organization of an odML structure remains completely flexible for the user. To compensate for the heterogeneity of the neuroscientific field, the odML framework allows the user to set the names, definitions and relations of Properties and Sections, as well as the data type, unit and uncertainty of the Value to her/his needs and conventions (reqt 4). The flexibility of the tree-like structure of odML-files makes it in addition easier to add over time new branches for, e.g., metadata resulting from a new preprocessing step of the data (reqt 5). The remaining requirements (6-8) were not or only in parts addressed by odML at the time I first used the odML metadata framework. To address these remaining requirements, I initiated a collaboration with the original developers of odML to optimize the framework. As result of this optimization process, the odML framework provides now besides a Python and Java API also a reduced, but functional Matlab library (reqt 6). Furthermore, the odML Python library provides now helper functions which can be used to find and extract metadata values with minimal user knowledge on the odML structure (reqt 7).

What the original technical publication of the software (Grewe et al., 2011) did not address, was to illustrate practical guidelines on how to create and use the metadata framework for complex experiments, especially for automatized data selection in an reproducible analysis workflow (reqt 8). In fact, there is a general failure in the field of neuroscience to properly advertise and support metadata management solutions for experimentalists who have little time to learn specifics of metadata standards (Tenopir et al., 2011). Paton (2008) even speaks of an "(o)*ver ambition* (of) *eager informaticians* (to) *design complex systems for capturing comprehensive descriptions of experiments that are time-consuming to populate and thus* (are) *never used*". To overcome this pitfall and improve the usefulness of the odML framework, in Chapter 3 I illustrate in a hands-on fashion how to generate an odML-file and outline a workflow to enter and maintain metadata in a corresponding odML-file collection for the example experiment I describe in Chapter 2. Furthermore, I demonstrate in a tutorial-like style how to access metadata from odML-files, not only to perform useful metadata analysis, but also to be able to integrate them as annotations in Neo, an open-source data framework that I will introduce in the next sub-chapter.

1.3 Data frameworks - making data access easy

To be able to share data, a proper documentation of the experiment including the compilation of a (digital) metadata collection as introduced in Section 1.2 is only the first step. As second step, it is necessary to provide easy access to the shared datasets and thus make the data and metadata usable or operational. This is unfortunately easier said than done. As already mentioned, in neuroscience the amount of available hardware as well as software tools by different vendors to acquire and analyse especially electrophysiological data is immense. Additionally many research laboratories use home-brewed devices and solutions when they set up an experiment. As a result, there is a large variety

[4] http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml

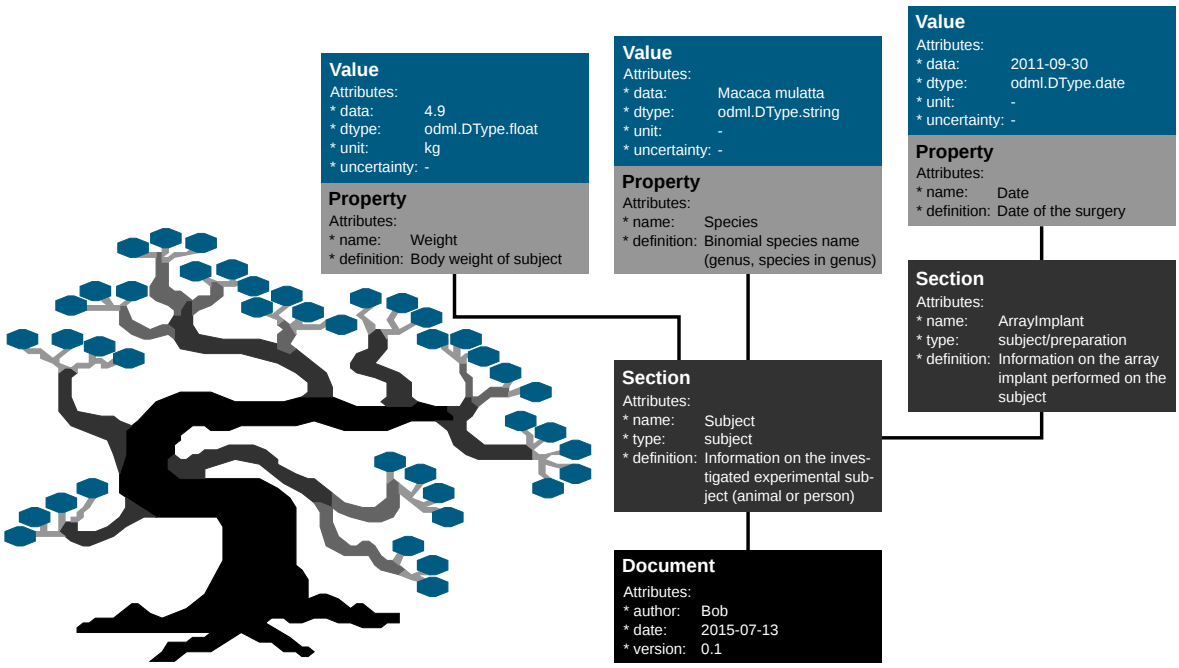

Figure 1.2 – The odML metadata framework: overview of the structural design. (figure 5 of Zehl et al., 2016) The displayed `Section`, `Property` and `Value` objects illustrate an exemplary subset of experimental metadata. A metadata item is stored in the `data` attribute of a `Value`. Additional attributes to define a `Value` are `dtype` (data type), `unit` or `uncertainty`. Although not displayed here, a `Property` can contain multiple `Value` objects. Which metadata are stored in the `Value` object(s) is defined by the `Property` via its `name` and `definition` attribute. `Property-Value/s` pairs that contain metadata of a similar context are grouped together into a `Section`. The context can be given by the `Section` attributes `name` and `definition`. If a sub-context needs to be defined in more detail, `Section` objects can be nested, i.e. containing sub-sections and thus form a tree-like structure. As root of the structure, the `Document` groups `Section` objects which are highest in the hierarchy. The attributes of a `Document` can state additional information on the odML-file, such as who created it (`author`), when it was created (`date`), or which odML framework `version` was used to create it.

of partially incompatible data models and corresponding file formats used within the neuroscientific community (Kuipers and van der Hoeven, 2009; Stoewer et al., 2014; Denker and Grün, 2015). Many of the existing data formats are even only accessible via proprietary software and only have limited support for metadata annotations (Stoewer et al., 2014). The zoo of file formats makes it not only particularly difficult for neuroscientists to set up generic and robust data loading routines, but also inhibits the development and usage of analysis methods across research projects (Garcia et al., 2014; Denker and Grün, 2015). As result, researchers spend a lot of time and effort on either converting data between different formats, or re-developing analysis methods that already exist for another format just to avoid the tedious conversion process (Garcia et al., 2014). Both strategies are often not rewarded by independent publications, but just serve the needs to be able to analyse data. This, again, has a strong negative influence on the willingness of research laboratories to share their data and corresponding loading or analysis routines with third parties. In addition, the process of data conversions usually approves the loss of information and is error prone. To promote and facilitate data sharing across the previously introduced levels A to D (cf. Section 1.1), general software solutions that guarantee easy access to any kind of experimental data and that allow the user to annotate data with metadata from standardized metadata frameworks (cf. Section 1.2) are highly needed (Stoewer et al., 2014; Teeters et al., 2015). In the last two decades, mainly three attempts were made to solve this demand (based on Garcia et al., 2014):

Approach 1) Development of a common file format that replaces all the currently available data formats. Examples for this approach are the European data format (EDF/EDF+[5], Kemp et al., 1992; Kemp and Olivan, 2003) for clinical electroencephalogram (EEG) and polysomnogram (PSG) data , the Neuroimaging Information Technology Initiative (NIFTI[6], Poldrack and Gorgolewski, 2014) for neuroimaging data, as well as the Neuroscience Information eXchange format (NIX[7], Stoewer et al., 2014) and the Neurodata Without Borders format (NWB[8], Teeters et al., 2015) for electrophysiological data.

Approach 2) Development of a language that provides a formalized description for each available data format which can be used to facilitate the conversion process between different formats. One example for this approach is the Signal Markup Language (SignalML[9], Durka and Ircha, 2004) which describes formats of biomedical time series (in particular EEG data).

Approach 3) Development of an application programming interface (API) that converts data from all formats into a uniform data framework. These data frameworks provide libraries that load data into an internal, uniform structure which can be used to access and analyse data independently of their original format. Examples for this approach are BioSig[10] (Vidaurre et al., 2011) and Neuroshare[11] which support a large number of biomedical (neurophysiological) file formats, NiBabel[12] which provides read and write access to neuroimaging file formats, and Neo[13] (Garcia et al., 2014) which provides several input/output (I/O) modules for electrophysiological file

[5] http://www.edfplus.info/

[6] http://nifti.nimh.nih.gov/nifti-1/

[7] http://g-node.github.io/nix/

[8] https://crcns.org/NWB

[9] http://braintech.pl/svarog/signalml/

[10] http://biosig.sourceforge.net/

[11] http://neuroshare.sourceforge.net/index.shtml

[12] http://nipy.org/nibabel/

[13] http://neuralensemble.org/neo

formats.

While a complete shift to common data formats would be desirable, the success of approach 1 mainly depends on the willingness of vendors to implement and provide the suggested formats for their data acquisition (DAQ) system. While the introduction of common file formats for biomedical and neuroimaging data was somewhat successful, the common file formats for electrophysiological data do not yet get the support they need by vendors to find acceptance in the community. As opposed to this, the success of the other two approaches is mainly driven by the community effort to develop the corresponding language descriptions (approach 2) or I/O modules (approach 3). The latter has the additional advantage that all I/O modules are designed to load the data into the defined internal uniform representation providing a base structure for software developers of interoperable analysis and visualization tools (Garcia et al., 2014).

For the electrophysiological example experiment described in Chapter 2, a commercially available DAQ system of a specific manufacturer was used which only provides data in its proprietary format. For this reason only a software solution of approach 3 came into question for setting up a robust and easy data access for all collaboration partners of the experiment and, in the end, the Neo data framework was chosen because of the following reasons:

Neo (from Neuroscience electrophysiology objects) is an object-oriented API for representing different electrophysiological data as objects in memory using I/O modules to read/write data from/to multiple file formats, which makes it robust and easy to extend regarding future developments and embeddings into more complex software components. The Neo API is implemented as an open-source, cross-platform Python package. A cross-platform operability was an important aspect in the decision for a data framework, because the computers used in the setup and by the collaboration partners may ran on both Microsoft Windows or Linux. Another reason to choose Neo was that the development of the I/O modules is driven by the community. Currently Neo supports over twenty file formats from different DAQ system manufacturers including Alpha Omega[14], Plexon[15], CED[16] (Spike2), and the one used in the example experiment, Blackrock Microsystems[17]. Additionally, Neo provides I/O modules for common file formats, such as the NeoMatlabIO for mat-files[18] and the HDF5IO for hdf5-files[19] which are both not only readable in Python, but also in Matlab. For the collaboration of the example experiment, this was an important functionality. The theoretical collaboration partners could analyse data in Python and exchange the processed data via a Matlab compatible format with the experimental partners. In addition to that, each I/O module of the Neo data framework reads typically accruing data of electrophysiological experiments into five standardized data objects (Figure 1.3):

`AnalogSignal` Data object representing one or multiple regular sampled, continuous analogue signals, such as a membrane potential recorded from an intracellular electrode, or four local field potential (LFP) signals recorded from the contacts of a tetrode. Besides containing the signal itself, `AnalogSignal` objects require the specification of the data units, the start time, and the sampling rate of the signal. In addition, the Neo framework recommends also defining a name and description identifying the signal, as well as stating the file-systems path or URL to the original data file.

[14] http://www.alphaomega-eng.com/

[15] http://www.plexon.com/

[16] http://ced.co.uk/

[17] http://blackrockmicro.com/

[18] https://www.mathworks.com/help/pdf_doc/matlab/matfile_format.pdf

[19] https://www.hdfgroup.org/HDF5/

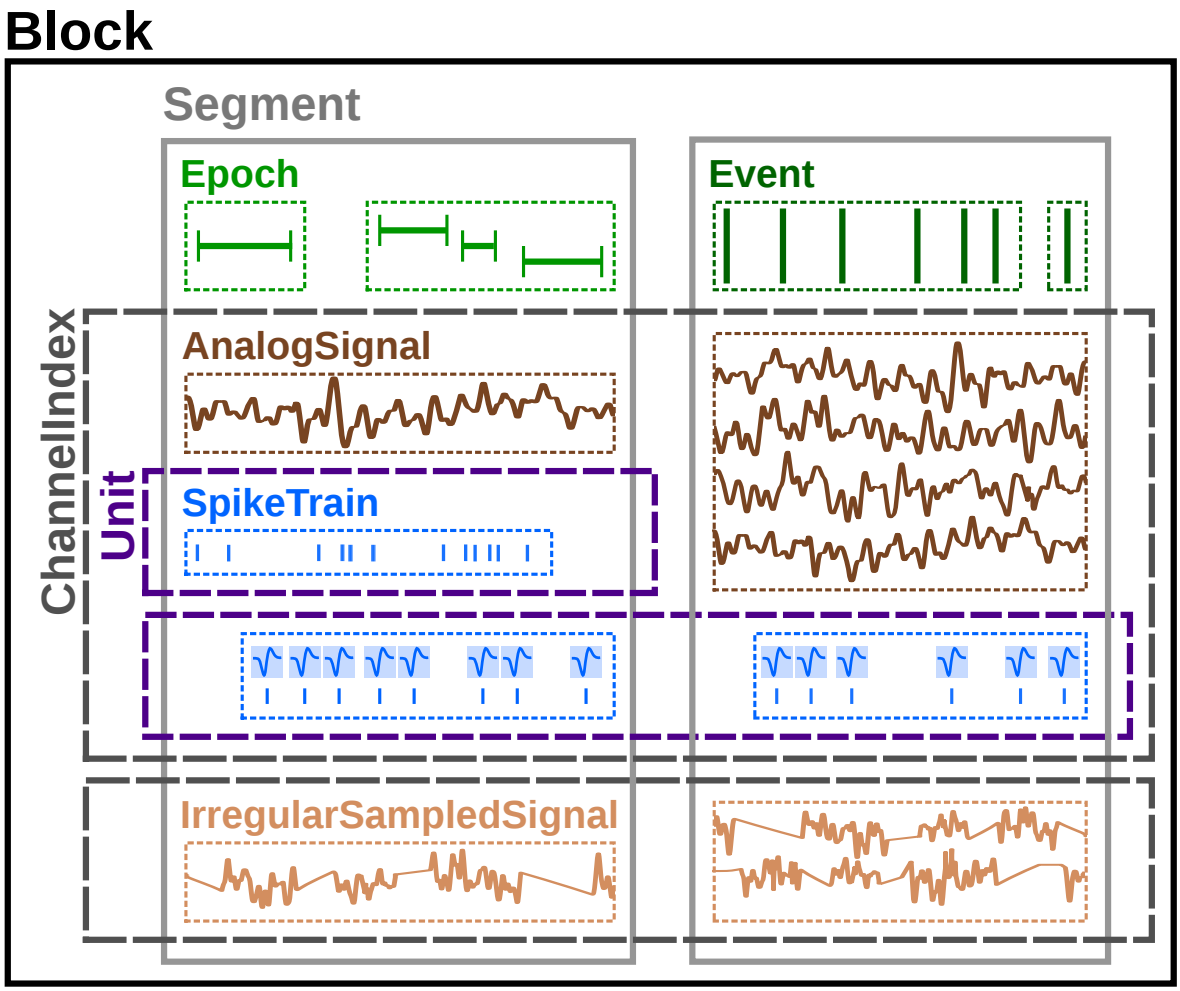

Figure 1.3 – The Neo data framework: overview of the Neo objects and their relations (based on figure 1, Garcia et al., 2014). One can distinguish between three types of objects: data objects (exemplary signals framed with thin dotted lines), grouping objects (framed with thick dashed lines), and container objects (framed with solid lines). The data objects cover with `Epoch` (light green), `Event` (dark green), `IrregularSampledSignal` (light brown), `AnalogSignal` (dark brown), `and SpikeTrain` (blue) including corresponding waveforms (exemplary signal in blue shaded small boxes) all relevant electrophysiological data types. These data objects are grouped into containers, where a `Segment` contains data objects that share a common clock (i.e. data that were recorded at the same time), and a `Block` in turn contains `Segment` objects that belong to the same experimental context. Grouping objects are then used to connect data objects of a `Block` across `Segment` objects. A `Unit` can be used to link `SpikeTrain` objects, and a `ChannelIndex` to link `AnalogSignal` and `IrregularSampledSignal`. In addition, a `ChannelIndex` can also be used to group `Unit` objects.

`IrregularSampledSignal` Data object that contains one or multiple irregular sampled, continuous analogue signals with their corresponding data time points and requires the specification of the data and time units of the signal. In addition, the Neo framework again recommends specifying a name, description, and the file origin of the data.

`SpikeTrain` Data object representing data of spiking activity, meaning the time points of action potentials recorded from one or multiple neurons (single or multi unit activity). In addition, `SpikeTrain` objects require the specification of the data units and the end time at which the spike data ended. The Neo framework also recommends specifying the start time of the spike data, besides the regular data object information (name, description, file origin of the data). Moreover, besides spike times, a `SpikeTrain` object can include an array of corresponding extracted waveforms. If such a waveforms array was attached, the Neo framework highly advocates also stating the sampling rate of the waveform signals, as well as the time shift between the start time of the waveform signal and the actual extracted trigger time point of the spike.

`Event` Data object that contains time points of single or multiple events, such as behavioural triggers. Besides the actual time points of the events, an `Event` object also requires the specification of labels for the corresponding events. In addition, the Neo framework also suggests adding the regular object information about the data (name, description, file origin of the data).

`Epoch` Data object representing time points and durations of single or multiple epochs, such as a stimulation period. As with `Event` objects, an `Epoch` object requires in addition the specification of labels for the corresponding epochs and supports the statement of the regular object information about the data (name, description, file origin of the data).

To establish the physical or temporal relation of these data objects, Neo provides additional grouping and container objects (Figure 1.3). There are two hierarchical organized container objects:

`Segment` Container object to contain all data objects which share a common clock, but do not necessarily have the same sampling rate, start or end time. Depending on the experimental context a `Segment` could reflect, e.g., a trial, a complete, or parts of an interrupted recording. To specify the experimental context, the Neo framework recommends stating the creation date and time of the original recording and the original file(s) of the contained data, besides the regular object information (name, description, file origin of the contained data). If several `Segment` containers are created, it is also highly recommended to provide an index to define their temporal order.

`Block` Container object that gathers all `Segment` objects with the corresponding data objects and all grouping objects for a given recording or set of recordings. To define the experimental context of a `Block`, the specification of the creation date and time of the original recording and the original file(s) of the contained data as well as the regular object information (name, description, file origin of the contained data) is recommended.

The two grouping objects contain references to data objects and are used to establish a relationship between them across `Segment` containers:

`ChannelIndex` Grouping object to represent a single or multiple recording channels via logical and/or physical indexing. It links `AnalogSignal` objects across multiple `Segment` containers and groups `Unit` objects in the case that the corresponding spike data of the `SpikeTrain` objects were obtained from the same recording channel (e.g., spike sorted units from data obtained from a single extracellular electrode). To further define the logical and/or physical context of the

`ChannelIndex`, it is recommended to also specify the name(s) and the logical or physical coordinate(s) of the recording channel(s), as well as the regular object information (name, description, file origin of the contained data).

`Unit` Grouping object that links `SpikeTrain` objects across multiple `Segment` containers if the corresponding spike data were emitted by the same single or multi unit (single or group of cells). The Neo frameworks recommends defining the regular object information (name, description, file origin of the contained data).

Besides the minimal metadata description provided by the stated required and recommended attributes of the Neo objects, it is also possible to associate additional metadata in key-value pairs, so called annotations. Both attributes and annotations can be used in filter functions provided by the Neo library to select data objects based on the associated metadata. This functionality combined with the object-oriented, standardized data representation make the Neo data framework particularly suited for integrating data into reproducible analysis workflows shared across laboratories. In fact, the Neo framework already promoted the development of subsequent software tools for common data analysis and visualization, such as OpenElectrophy[20](Garcia, 2009), SpykeViewer[21](Pröpper and Obermayer, 2013) and Elephant[22](Denker et al., 2015a,b). In addition, the German Node (G-Node) of the International Neuroinformatics Coordination Facility (INCF) has integrated the Neo data framework in combination with the odML metadata framework into their data management platform[23], providing, thus, a centralized system for sharing electrophysiological data and metadata (Garcia et al., 2014; Sobolev et al., 2014b,a).

As for many other management solutions, the task of implementing the software into ones own workflow remains with the scientists. For the example experiment this included the following two tasks. The pre-existing Blackrock I/O module of the Neo data framework was unfortunately not sufficient to load the corresponding experimental data, making it necessary to extend and optimize the available code accordingly (task 1, see Section 2.5.2.1). The re-implementation was designed to optimize the general Blackrock I/O module and could therefore be published in the new software release of the Neo data framework. To connect the data framework with the metadata framework, it was furthermore necessary to implement an experiment-specific I/O module that integrated additional data annotations with metadata from the corresponding odML metadata framework (task 2, see Section 2.5.2.2). Although the accomplishment of both tasks was time consuming, the effort to integrate a more standardized data and metadata management enabled the publication of data and metadata including the provision of reliable access routines that guarantee more comprehensive data and metadata queries, and the inclusion of data in analysis workflows in a more robust and reproducible manner inside, but also outside the research collaboration of the example experiment.

1.4 Putting the concepts to the test... (thesis overview)

In the central part of this thesis, I will detail the concepts of data sharing by demonstrating how to implement and use a metadata and data framework for an electrophysiological example experiment with behaving monkeys that originated from an interdisciplinary collaboration between the CoMCo

[20] https://github.com/OpenElectrophy/OpenElectrophy

[21] https://spyke-viewer.readthedocs.io/en/latest/

[22] http://neuralensemble.org/elephant/

[23] https://portal.g-node.org/data/

group at the INT of the Aix-Marseille University in France and the SN group at the INM-6 of the Jülich Research Centre in Germany (for abbreviation, cf. beginning of Chapter 1).

To provide the necessary foundation for my thesis, in Chapter 2 I will first describe the complete example experiment, following the guidelines of Scientific Data, an open-access, peer-reviewed journal for publishing scientifically valuable datasets. This data descriptor, includes a profile for all monkeys used for this project (see Section 2.2), a documentation of the task and daily routines the animals had to accomplish with all possible behavioural conditions (see introduction of Chapter 2 and Section 2.1), a full description of the experimental setup including all hard and software components of the experimental apparatus, the behavioural control system, and the neural recording platform (see Section 2.3), as well as a summary of required technical validations of the data (see Section 2.4) in form of (pre-analysis) preprocessing steps (e.g., as the offline spike sorting described in Section 2.4.3) or quality assessment procedures (e.g., as the electrode and trial rejection described in Section 2.4.4.3). I will also introduce the scientific potential of the corresponding experimental data which led to the decision to publish two specific datasets of the example experiment in Scientific Data (see Section 2.6) and explains the scientific background of an example publication (to be submitted in Neuron) of research findings acquired by the collaboration partners (see below, and Chapter 4).

Both publications (data and research findings) benefit from the implementation of the odML metadata and the Neo data framework (for an introduction to the frameworks see Section 1.2 and Section 1.3, respectively) and their integration into the daily analysis workflow of the collaboration partners (see Section 2.5 and Chapter 3). In this context, I summarize in Section 2.5.1 the preparatory steps I needed to take to implement the odML metadata framework for the example experiment and demonstrate in Section 3.1 why, especially for the long term, the invested work and time is worth the effort. In Chapter 3, I will show for the case of the example experiment how an odML metadata file collection can be implemented (see Section 3.2) and explain several functionalities for the practical usage of the odML framework (see Section 3.3). In Section 2.5.2, I will summarize the base implementation of the Blackrock I/O module for the Neo data framework (see Section 2.5.2.1) and illustrate how to further annotate Neo objects with metadata from an odML-file collection in an experiment specific loading routine (see Section 2.5.2.2).

In Chapter 4 I demonstrate the usage of the implemented Neo data and odML metadata framework for the analysis of the example experiment, completing the main part of the thesis with the presentation of a research publication on spectral properties of motor cortical local field potential (LFP) signals and their characterization into emergent spatio-temporal patterns in context of selected behavioural conditions of the example experiment.

In Chapter 5 I resume the difficulties of setting up a metadata and data framework for such complex experiments, and summarize the implemented solution of the example experiment. In this context, I briefly discuss alternative approaches and give an outlook on how to complete the data and metadata management and corresponding analysis workflow towards full reproducible research publications.

Chapter 2

Descriptor of the example experiment

The following chapter is in mainly based on text parts of the manuscript "Massively parallel multi-electrode recordings of macaque motor cortex during an instructed delayed reach-to-grasp task". The manuscript will be submitted to a data journal (e.g., Scientific Data).

The following chapter provides a data descriptor of the electrophysiological example experiment. A data descriptor is a specific publication format of open-access, peer-reviewed data journals, such as Scientific Data, that provides a citable documentation for scientifically valuable datasets which must be made available on a public, community-recognized repository. The advantage of such a certified data descriptor is not only that the authors are rewarded for sharing their data by means of a citable peer-reviewed publication, but also that they gain a certain protection against data misuse by third parties. The journal Scientific Data dictates that a data descriptor should provide a complete description of the experiment including the experimental background and design, the experimental setup, the workflow to produce and computationally further process the experimental data, as well as a summary of the content of the published datasets and any technical validation needed to guarantee the quality of the data. Besides a written documentation, it is also required to compile the described information into a machine-readable metadata format, such as the chosen odML format for the example experiment (cf. Section 1.2, Section 2.5.1, and Chapter 3). In addition, it is recommended to provide access to any implemented code to generate, load or process the data, such as the loading routines I implemented for the data of the example experiment based on the Neo data framework (cf. Section 1.3 and Section 2.5.2). With all these journal requirements, the construction of a data descriptor is well suited as guideline for a proper documentation of an experiment which corresponding data are supposed to be shared not only, as intended by the journal, with unknown third parties (cf. level D, Section 1.1), but also within a laboratory or a research collaboration (cf. level A to C, Section 1.1). In the following chapter, I will demonstrate the magnitude of such a data descriptor by providing a documentation of the complete example experiment and explain at the end why only two datasets are currently prepared for a publication in a data journal, such as Scientific Data. However, before I will go into detail, let me start with a brief overview of the example experiment to introduce its origination process, the general course of the project, and its scientific context.

The example experiment originated from an interdisciplinary collaboration between the experimental CoMCo group at the INT of the Aix-Marseille University in France and the theoretical SN

group at the INM-6 of the Jülich Research Centre in Germany. The actual experiment was developed and conducted in the CoMCo group in Marseille with two permanent researchers (Dr. A. Riehle and Dr. T. Brochier) and at least six secondary labour force (Dr. Y. Hao, Dr. G. Prabhu, as well as the PhD students M. Zaeppfel, M. Duret, and T. Milekovic, and the master student S. Wirtssohn) working on the project for the last years. The resulting datasets of the experiment were unrestrictedly shared with the SN group in Germany to develop and perform complex analysis methods on the corresponding data in close collaboration with the members of the experimental group (data sharing level C, cf. Section 1.1). On the side of the theoretical group, two permanent researcher (Prof. Dr. S. Grün and Dr. M. Denker) and at least five secondary labour force (Ph.D. students E. Torre, L. Zehl, V. Rostami, J. Sprenger, and P. Qualigio) were or are involved in the project.

In the example experiment, high-dimensional and multi-scale electrophysiological datasets (for details see Section 2.3.3) were recorded from a monkey motor cortex during an instructed delayed reach-to-grasp task (for details see Section 2.1.1), using a chronically implanted multi-electrode array (for details see Section 2.2.2). The main objective of the experiment was to explore the neuronal signals related to hand position and force that are processed in the motor cortex during the preparation and execution of a grasping movement. The resulting new scientific insights can be used, for example, to optimize the decoding strategy in brain-machine interfaces for hand movement control (cf. Milekovic et al., 2015). Within the collaboration and in corresponding research publications (Riehle et al., 2013; Milekovic et al., 2015; Torre et al., 2016; Zehl et al., 2016), the experiment was consequently titled as *reach-to-grasp* or *R2G* experiment. Besides the standard implementation of the reach-to-grasp task, also a couple of modified and different task paradigms were conducted (for details see Section 2.1.2). In total, the experiment comprises datasets recorded from three monkeys (monkey L, T, and N, for details see Table 2.1). A fourth monkey (monkey E) is currently in training and therefore not further described in this thesis, except for the first basic information given in Table 2.1. All monkeys were recorded in the same setup, with small differences due to updates of some hard and software parts (for details see Section 2.3).

All monkeys (with one exception) conducted subsequently underwent one recording period for each hand with a multi-electrode array implanted in the corresponding contra-lateral brain hemisphere (for details see Table 2.1). A recording period started about one week after the multi-electrode array was implanted and ended when the quality of the recordings dropped below a useful level (for details see Table 2.1). Usually, a single recording period lasted for about 6 months, with recordings conducted on regular workdays. Weekends as well as holidays were usually excluded. A typical recording day consisted of four steps: (1) taking the monkey out of the cage, (2) placing the monkey in a primate chair in front of the experimental apparatus (for details see Section 2.3.1), (3) conducting several recordings of the neuronal activity (for details see Section 2.2.2 and Section 2.3.3) while the monkey performed a task (for details see Section 2.1 and Section 2.3.2), and (4) returning the monkey back to its cage.

The number of performed recordings during a single recording day depended on the motivation of the monkey, with a single recording lasting usually for 10 minutes (for averages see Table 2.1). Typically, a recording day ended after one hour (for averages see Table 2.1). A single recording is defined by all recorded signals that share a common clock and are stored in a dataset consisting of multiple files (for details see Section 2.3.3). During a single recording period a monkey performed on average 300 recordings (for details see Table 2.1). In the course of the R2G experiment, all performed animal procedures were approved by the ethical committee of Aix-Marseille Université (authorization A1/10/12) and conformed to the European and French government regulations.

		monkey T	monkey L	monkey N	monkey E
	gender	female	female	male	female
	date of birth	2003-05-28	2004-03-15	2008-05-15	2010-04-29
first period	**body weight (pre-training)**	4.3 kg	3.6 kg	4.2 kg	6.8 kg
	training start	2007-02-26	2008-??-??	2012-05-??	2015-08-03
	body weight (pre-surgical)	4 kg	5 kg	4.9 kg	-
	surgery (implant)	2010-02-26	2010-09-15	2013-05-01	-
	hemisphere	left	right	left	right
	array (serialno)	1025-0443	1025-0503	6250-0810	-
	file identifier	t	l	n	e
	first recording	2010-03-09	2010-09-30	2013-05-06	-
	last recording	2011-02-11	2011-04-15	2013-11-29	-
	# recording days	101	57	131	-
	<recording day dur>	40 min	60 min	40 min	-
	# recordings	366	308	432	-
	<recording dur>	9 min	10 min	11 min	-
	surgery (removal)	2011-05-17	2011-06-23	2014-05-30	-
second period	**training start**	2010-11-02	-	2013-09-13	-
	body weight	5.3 kg	-	6.6 kg	-
	surgery (implant)	2011-05-17	-	2014-05-22	-
	hemisphere	right	-	right	-
	array (serialno)	2717-0573	-	8596-001139	-
	file identifier	a	-	i	-
	first recording	2011-05-30	-	2014-06-04	-
	last recording	2011-10-04	-	2015-02-26	-
	# recording days	55	-	139	-
	<recording day dur>	20 min	-	30 min	-
	# recordings	87	-	349	-
	<recording dur>	10 min	-	9 min	-
	surgery (removal)	2011-??-??	-	2015-02-26	-

Table 2.1 – Reach-to-grasp monkeys: dates and facts. For the general understanding of the table: "#" stands for "number of", "dur" stands for "duration", and "<*>" stands for "averaged". Remaining "?" in the table reflect metadata, which could not be retrieved so far. Things to note: Monkey T was the first monkey trained in the reach-to-grasp experiment, followed by monkey L, and after her, with a delay of one year, monkey N. Monkey E is currently trained in the task using the left hand. Monkey E's surgery to implant the array will probably take place early next year (2017). The first training periods of monkey T and L are with 3 and 2 years relatively long compared to monkey N and E with roughly only 1 year. However, the first training of monkey T and L started with a preliminary version of the task setup and was additionally interrupted due to a relocation of the laboratory. If a second recording period was planned for the monkey, the removal of the first implanted array, or to be precise the removal of the corresponding array connector, was performed during the implant surgery of the new array into the motor cortex of the other hemisphere. The second training of monkey T started before the first array was removed to enable recordings from the activity of the ipsi-lateral hemisphere during three weeks in November 2010 when the retraining for the left hand already started. The listed last recording was solely conducted to perform another recording under the mapping paradigm (cf. Section 2.1.2). The last recording of the contra-lateral hemisphere of the first recording period of monkey T was performed on October 15, 2010.

2.1 The task(s)

In the course of the R2G experiment, the paradigm of the standard reach-to-grasp task was sometimes modified or changed. These modifications and changes are important for the correct interpretation of the corresponding datasets and sometimes even influence the way the data need to be accessed. In the R2G experiment the standard task paradigm needs to be distinguished from eight other paradigms. Note that in all monkeys, the file name of each dataset is (usually) composed of the monkey's file identifier (t, a, l, n, or i, cf. Table 2.1), the date of the recording day (using the format *ddmmyy* for datasets of the first recording period of monkey T and the format *yymmdd* for all remaining datasets), and the identifier of the recording (e.g., -001, for the first recording of the day). Thus, the file names typically do not provide information on the chosen task paradigm. Furthermore, the task paradigms are not or only indirectly distinguishable from each other via information provided in the data files. For this reason, it is crucial to provide the corresponding metadata elsewhere to avoid confusion, misinterpretations and misuse of the data. In the R2G experiment, the information on which task paradigm was executed with which settings in which recording was primarily registered in the laboratory notebooks, and, at least for most datasets of monkey L and T, stated in recording-specific spreadsheets. In the following sub-sections, I will provide a description of all possible task paradigms even if they were only conducted once or a few times in the course of the experiment.

2.1.1 Two-instructions paradigm (standard task)

The monkey's standard task is defined via the two-instructions (2-inst) paradigm which can be described as follows. Starting from a neutral home position (cf. Figure 2.1), the monkey had to grasp a cubic object using either a precision grip (PG) or a side grip (SG).

The PG had to be performed by placing the tips of index and thumb in a groove on the upper and lower sides the object, respectively (cf. Figure 2.1). For SG, the tip of the thumb (T)) and the lateral surface of the other fingers (e.g., middle finger) had to be placed on the right and left sides of the object (cf. Figure 2.1). The monkey had then to pull out the object along a horizontal shuttle against one of two possible loads requiring either a high or low pulling force (HF and LF, respectively) and hold the object in a broad position window for a certain amount of time before returning again to the home position. In the course of such a trial, the instructions on the requested grip and force type were provided to the performing monkey independently through two consecutive visual cues which were separated by a one second delay (cf. Figure 2.1). The visual cue system was positioned above the target object and consisted of five LEDs arranged in a five-pip cube of which the illumination of a specific set of two corner LEDs coded for the grip and force type (cf. Figure 2.1). As a result, from the possible combinations and scheduling of the instructions on the grip and force types, the monkeys had to learn to correctly interpret eight different trial types (cf. PG-HF, PG-LF, HF-PG, LF-PG, SG-HF, SG-LF, HF-SG, LF-SG in Figure 2.1). For a single recording, a subset of these trial types was chosen to be presented in a sequence of 200 trials on average. To simplify referencing which combination of the trial types were chosen in a recording, the possible subsets were coded as integer numbers, called recording conditions (cf. Table 2.2).

In general, the pool of possible recording conditions can be divided in grip-cue first and force-cue first trial sequences (cf. Figure 2.1). Mixing grip-cue first trial types with force-cue first trial types within the trial sequence of one recording resulted usually in a drop of the monkey's performance and was therefore avoided. As opposed to this, the conditions of recordings during one recording day were often alternated, to keep the motivation level of the performing monkey up. Moreover, for all recording conditions including more than one trial type, it was additionally possible to alternate

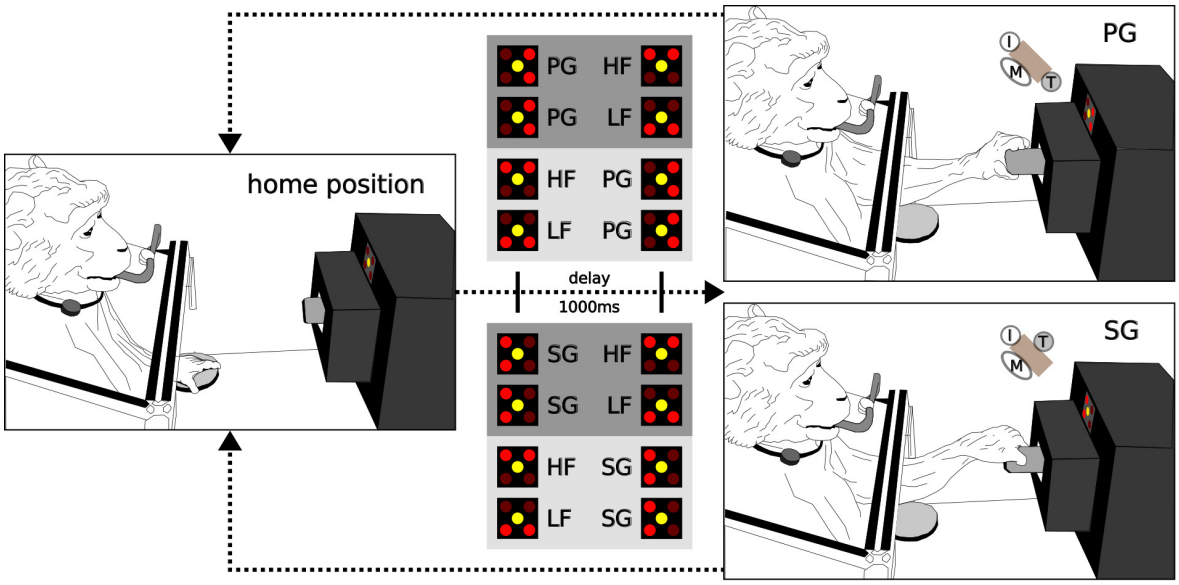

Figure 2.1 – Schema of the standard reach-to-grasp task. (modified figure 3, panel a of the unpublished manuscript for Brochier et al., prep) In each trial of a standard reach-to-grasp task the monkey started from a neutral home position as illustrated in the left sketch. The monkey then had grasp, pull and hold a tilted cubic object using either a precision grip (PG) or a side grip (SG) as shown in the top and bottom right sketch, respectively. The inset of top and bottom right sketch indicates the actual position of the index finger (I), the thumb (T), and the middle finger (M) on the object (brown cube) for a precision and side grip, respectively. In addition to the grip type, the monkey had to adjust its pulling force to two possible object loads (low force: LF, and high force: HF). In the course of each trial, instructions on the requested grip and force type were provided to the performing monkey in two consecutive visual cues which were separated by a one second delay. The middle column of the figure illustrates which illumination of corner LEDs of the visual cues system coded for PG, SG, HF and LF, and lists all eight trial types (PG-HF, PG-LF, HF-PG, LF-PG, SG-HF, SG-LF, HF-SG, LF-SG) resulting from the combinations of all grip with all force types as well as the alternation of their presentation order. Trial types which first give instructions on the grip type (grip-cue first) are shaded in dark grey, while trial types first providing the force type (force-cue first) are shaded in light grey. As indicated by the arrows, the monkey had to return back to the home position after each successful or unsuccessful trial. The combination of trial types occurring in a trial sequence as well as how the selected trial types alternate (random or in blocks) defines the task condition of a recording. Table 2.2 provides an overview of possible task conditions in the reach-to-grasp task.

	cnd code			first cue (CUE) type		first cue (CUE) seq		second cue (GO) type		second cue (GO) seq		# trial types
grip-cue first	1			SG	PG	ra	bl	LF	HF	ra	bl	4
	1	1		SG	PG	ra	bl	LF			(bl)	2
	1	2		SG	PG	ra	bl		HF		(bl)	2
	1	3		SG			(bl)	LF	HF	ra	bl	2
	1	4			PG		(bl)	LF	HF	ra	bl	2
	1	3	1	SG			(bl)	LF			(bl)	1
	1	3	2	SG			(bl)		HF		(bl)	1
	1	4	1		PG		(bl)	LF			(bl)	1
	1	4	2		PG		(bl)		HF		(bl)	1
force-cue first	2			LF	HF	ra	bl	SG	PG	ra	bl	4
	2	1		LF			(bl)	SG	PG	ra	bl	2
	2	2			HF		(bl)	SG	PG	ra	bl	2
	2	3		LF	HF	ra	bl	SG			(bl)	2
	2	4		LF	HF	ra	bl		PG		(bl)	2
	2	1	3	LF			(bl)	SG			(bl)	1
	2	1	4	LF			(bl)		PG		(bl)	1
	2	2	3		HF		(bl)	SG			(bl)	1
	2	2	4		HF		(bl)		PG		(bl)	1

Table 2.2 – Possible conditions of the standard reach-to-grasp task. A reach-to-grasp task condition is coded as number of one, two or three digits (cnd code). This cnd code was invented by the collaboration partners to simplify the identification of the corresponding recordings. For each cnd code, the first digit is always set. It is set to 1 for grip-cue first conditions, where the first cue (CUE) reveals the grip type (SG or PG) and the second cue (GO) reveals the force type (LF or HF), and set to 2 for force-cue first conditions, where the order of revealing grip and force type is reversed. The second and third digit can be set to 1 or 2 and/or 3 or 4 if the combinations of trial types was limited to LF or HF and/or SG or PG trials, respectively. The number of (#) possible trial types for each condition results from the possible combinations and scheduling of the instructions on the grip and force types (cf. Figure 2.1). If grip and force type are both restricted, only one trial type is presented to the monkey in the course of one recording. In such a case, the order of second and third digit of the cnd code reflects also the order of grip and force type instructions (e.g., grip type is coded on second and force type on third digit for grip-cue first trial types). For each condition it can be further decided if the sequence (seq) of grip and force types (gt and ft) alternate randomly (ra) or between blocks (bl) of a defined number of trials (e.g., 1 or 10). If only one grip or force type occurs in the trial type combinations of the chosen condition, the declaration of the sequence is redundant and therefore stated in brackets, because the corresponding type is repeated over the course of the whole recording.

the sequence of grip and/or force types randomly or in blocks of different size. Details on how the task, the trial scheme and the corresponding behaviour of the monkey were controlled are stated in Section 2.3.2.

2.1.2 Other tasks

In the course of the whole R2G experiment, the basic layout of the 2-inst paradigm was alternated in five different ways. The first alternative can be called **observation (obs) paradigm**. For the corresponding recordings the monkey was, instead of performing the task itself, supposed to watch a human performing the 2-inst paradigm. The observation paradigm paradigm was conducted only once with monkey L (recording l110208-005), and can therefore be neglected for any further data analysis of the project. The second alternative can be called **ipsi-lateral-two-instructions (ipsi) paradigm**. As the name suggests the task was exactly the same, but the neuronal activity was recorded from the ipsi-lateral instead of the contra-lateral motor cortex. This paradigm was tested with monkey T and N after the first recording period, during a short period of the second training for the other hand (cf. Table 2.1). The third alternative can be called **grasp-orientation paradigm**. In the corresponding recordings the monkey was forced to use different wrist orientations when grasping the

object. For this, the trial scheme matched the condition 1 setting of the 2-inst paradigm, but the object was rotated between blocks of trial. The idea behind this paradigm was to explore if, how and where on the array the brain activity coding the grip is modulated by the wrist orientation. This paradigm was only tested with monkey N, but will likely be tested again in monkey E. The fourth alternative can be called **isometric-contraction (iso) paradigm**. This paradigm was also only tested with monkey N, but again, will likely be tested further with monkey E. The trial scheme also matched the condition 1 setting of the 2-inst paradigm, but the monkey had to pull a fixated object. The goal of this paradigm was to enforce a isometric instead of a isotonic muscle contraction and see if the corresponding brain activity differs. The fifth alternative was called **one-instruction (1-inst) paradigm**. In this paradigm the monkey had to perform in principle the same behavioural task as in the 2-inst paradigm: the monkey received the instruction to use one of the two grip types in the first cue (CUE) and then again as repetition in the second cue (GO). The object load was not revealed, so that the monkey had to adjust the pulling force to the randomly set or in blocks alternating object load during the movement. In total, the 1-inst paradigm was conducted 53 times with monkey L and 31 times with monkey T. With monkey N this paradigm was not tested. For selecting data, the recordings with the 1-inst paradigm are only indirectly distinguishable from recordings with the standard paradigm by means of a repetition of the grip instruction in the course of each trial instead of providing information on the force type. Under consideration of the definition of conditions in the standard paradigm, recordings using the one-instruction paradigm can be determined as condition 133 for an instruction repetition of SG and 144 for an instruction repetition of PG in the recorded trials (cf. Table 2.2).

To be able to relate the neural recordings and subsequent analysis results more closely to the actual motor output and sensory feedback covered by the array implant in the motor cortex, a complementary **mapping (map) paradigm** was conducted in all monkeys. In the corresponding recordings the activity recorded from each electrode of the implanted array was analysed in respect to its representation of specific body parts (cf. Riehle et al., 2013). In monkey L and T this was tested passively by touching and moving the corresponding contra-lateral upper limb during a recording, which reflects the sensory feedback representation of the corresponding body parts. Due to an update and extension of the used hardware, it was possible to use a stimulation protocol on monkey N to generate a map of the motor output. For this, each electrode was every 2 seconds stimulated with a train of 15 biphasic pulses with $0.5\,ms$ duration. The resulting evoked movement in the contra-lateral upper limb, or other body parts (face, trunk) was observed and manually documented. The stimulation intensity was initially set to $20\,\mu A$ for each electrode and then gradually increased to find the minimal intensity needed to evoke a motor response. If, however, a motor response could not be evoked with $100\,\mu A$, the corresponding electrode was considered unresponsive.

In contrast with the other paradigms, the **sleep paradigm** occurred by accident. During a few recording days monkey T and L fell asleep, providing the opportunity to record the monkey's brain activity. The sleeping behaviour was recorded on 11 and 5 recording days for monkey T and L, respectively. In monkey N it was instead possible to record by chance a **resting-state (rest) paradigm**. On two recording days during the second recording period it was possible to record a resting-state behaviour of the monkey (i140615-002 and i140701-004). Both datasets include a video showing the monkey more or less motionless, but not sleeping, in the primate chair.

The last paradigm, the **single pulse micro-stimulation and recording (stim) paradigm**, was also so far only feasible in monkey N, because of the already mentioned update of the used hardware. Monkey E will very likely provide more datasets with this paradigm. The goal of the stim paradigm was to determine via two different stimulation protocols the structure of the underlying network of

neurons recorded by the electrodes of the array. In the first protocol, one electrode was stimulated, while the possible response was recorded from all other electrodes. In the second protocol, the response of one electrode was recorded, while all other electrodes were subsequently stimulated.

2.2 The subjects

The reach-to-grasp experiment was designed for Rhesus macaques (*Macaca mullata*). The fact that the subjects are living, intelligent and social animals, is already a cause for variability in the data which cannot be easily explained or reproduced. For this reason, it is of utmost importance to document as much information as possible about each subject in the course of the experiment. This provides at least the chance of relating general differences in the recorded data to general differences in the constitution, behaviour, and performed surgical procedures of each monkey. Besides this rather complex reason to provide a proper documentation about the participating subjects, it is in any case necessary for each member of the collaboration team to be able to identify the subjects and know about their particularities in behaviour and neuronal recordings to be able to select meaningful data for corresponding analysis procedures. For the R2G experiment the following general subject information needed to be shared.

Monkey T and L are females, while monkey N is male. All monkeys, except for monkey L, subsequently underwent one training and recording period for each hand. Monkey L had one phalanx missing on the right thumb and was therefore only able to perform the task with the left hand. Recorded files can be identified by the first and second letter of the corresponding monkey name for the first and second recording period, respectively. With this, the identifiers are “l” for monkey L, as well as “t” and “a” for monkey T, and “n” and “i” for monkey N for the first and second recording period, respectively. Correct behaviour of the monkeys was rewarded with apple sauce. To increase the attraction of this food reward, the three monkeys were only fed with dry food in the home cage, except during weekends and holidays, where the food was supplemented with fruits and vegetables. For all three monkeys, the water access was always unrestricted.

In the following sub-sections, I will describe in more detail the general training and performance, as well as surgical protocols, and differences of the implanted arrays and corresponding recordings for each monkey. In addition, Table 2.1 gives an overview of all important dates and facts for all current monkeys of the R2G experiment.

2.2.1 The training and general performance

The first training period started for each monkey with getting accustomed to the experimenter (Dr. T. Brochier or the supporting post-doc/student) and the primate chair. After this settling period each monkey had to master the standard reach-to-grasp task. For this, each monkey first had to learn the correct sequence of trial events and the corresponding visual instructions for the SG-LF trial type. Precision grip trials were then introduced by restraining the finger placement onto the top and bottom FSR sensors using guiding blocks around the object. If the monkey grasped correctly, it had to pull and hold the object to receive a reward. The time point to deactivate the reward pump was set proportional to the duration of holding the object in a broad position window, with a maximum amount of reward for the required holding period of $500\,ms$. With this mechanism, all monkeys rapidly learned to grasp, pull and hold the object for the correct duration in nearly all trials. As soon as the grip performance of the monkey was reliable, it had to learn to adjust the pulling force between two different object loads (LF and HF) as instructed by the corresponding visual cue. The training

of each monkey was completed after on average 80 % of trials were performed correctly for grip and force first conditions. Nevertheless, it was necessary to carry out some additional training during the recording periods of the monkeys. To push the monkeys to anticipate the object load according to the given visual cue, the allowed time to pull the object in the position window was subsequently decreased over a couple of recordings on four recording days for monkey T (2010-09-16/17/27/29) and on three recording days for monkey L (2010-11-09/10/29). In monkey L it was also necessary to perform a training against her chewing movements during the trials, because the analogue signals of the multi-electrode array were contaminated by huge chewing artefacts. For this, during a recording, the analogue signal of a recording channel that did not show any spiking activity was copied and fed back into the behavioural control system (cf. Section 2.3.2). There the chewing artefact was used on the one hand as sound feedback for the monkey producing a loud white noise, and on the other hand as trigger (detected via a threshold) to abort a trial before a reward was given. Monkey L learnt very quickly the relationship between her chewing, the sound feedback and the trial cancellation, so that only one recording day (2010-11-22) was necessary to train her accordingly.

In general, the training period with the first hand took longer than the retraining with the opposite hand (i.e., second training period), because it included the settling period of the monkey, and also because the monkey learned about the task for the first time. Additional variations in training durations are caused by changes or breaks in training protocol and each monkey's general learning ability. Differences in the latter are usually also reflected in the performance during the recording periods of each monkey. In summary, the monkeys learning and performance abilities were described by the experimenters in the following way. For monkey T and L, especially the first training was long (around three and two years, respectively), which was caused by a change of the training protocol as well as a training break due to a relocation of the experimental laboratory. The training protocol was changed, because both monkeys started to work on a preliminary version of the task setup. Adapting to the actual reach-to-grasp standard task took time for both of them, but was especially difficult for monkey T. In this context, the experimenter described monkey T as being too quiet, slow and not a hard worker. This was also reflected in the monkey's performance during the recording periods. The experimenter mentioned that monkey T took often long breaks during a recording day or simply did not work at all which is evident in the average number of recordings per day of only 3.6 during the first and 1.5 during the second recording period. Despite the delays caused by the described reasons, the training with monkey L was easy and fast. She rapidly habituated to the training chair in the first training period and could start the task training within a month. Generally, the experimenter described monkey L as being eager to work, quick and efficient during the task which is reflected on average by 5.4 recordings per day during her recording period. Unfortunately she was also nervous which resulted in a relatively high number of trial errors caused by a too early movement onset[1]. She also did not like to be manipulated which affected the procedure during the mapping-task. Monkey N was calmer than monkey L, but overall less motivated. He learned the task at a much slower pace, although this is not reflected in the duration of the first training period (roughly a year), because he did not face the changes in the task setup nor did he have a break in the training period as the first two monkeys did. The performance ability of monkey N is reflected in the average number of only 3.3 and 2.5 of recordings per day during the first and second recording period, respectively. In general, he was less attentive, leading to high numbers of grip errors, especially for force-first recordings, and fewer numbers of force-first recordings in general[1].

[1]This statement is mainly based on a verbal message of the responsible experimenter.

2.2.2 The array: implant, recording quality, removal and location

After a training was completed, a surgery was scheduled to chronically implant a 100-electrode Utah-Array (Blackrock Microsystems, Salt Lake City, UT, USA) in the motor cortex contra-lateral to the trained hand. Which hemisphere was implanted first and second for which monkey is stated in Table 2.1. All implanted arrays consisted of one 10-by-10 electrode grid with 96 active Iridium oxide electrodes. The electrode length was $1.5\,mm$ with an inter-electrode distance of $400\,\mu m$. The active electrodes had on average an industrial impedance of $50\,k\Omega$ and were connected, along with one ground and two references, through a wire bundle of $4\,cm$ length to a high-density CerePort Connector.

The surgery, as first described in Riehle et al. (2013), was performed by a veterinary surgeon under deep anaesthesia using full aseptic procedures. Anaesthesia was induced with $10\,mg/kg$ of intra-muscular ketamine and maintained with $2-2.5\,\%$ isoflurane in $40:60\,O_2-$air. To prevent cortical swelling, $2\,ml/kg$ of intravenous mannitol infusion was slowly injected over a period of $10\,min$. A $20\,mm\,x\,20\,mm$ craniotomy was performed over the motor cortex and the dura was incised and reflected. The array was inserted using a pneumatic inserter (Array Inserter, Blackrock Microsystems) and covered with a sheet of an artificial non-absorbable dura (Preclude, Gore-tex) to avoid attachment of the array to the dura. The real dura was sutured back and covered with a piece of an artificial absorbable dura (Seamdura, Codman). The bone flap was put back at its original position and attached to the skull by means of a $4\,mm\,x\,40\,mm$ strip of titanium (Bioplate, Codman). The array connector was fixed to the skull on the opposite side with titanium bone screws (Bioplate, Codman). As the last step of the surgery, the skin was sutured back over the bone flap and around the connector. The monkey received a full course of antibiotics and analgesics before returning to the home cage. After a week of resting, the recording period started (cf. dates in Table 2.1).

The quality of the recorded data, especially the spiking activity, depended in large parts on the quality of the implant procedure of the array, i.e. how symmetrically the array was inserted into the cortex. In monkey T, unfortunately both implant procedures did not go smoothly resulting in poor recordings of the spiking activity in particular (cf. low numbers of identified single and multi units in Table A.6 and Table A.7). In addition, monkey T developed health issues during the second recording period. To protect the monkey's health the second recording period was terminated after four months. It was decided to not consider any recording from these datasets for further analysis, because it was unclear if the health issues affected the data. The second recording period will, therefore, not be further described in this thesis. Also the first recording period of monkey N was rather disappointing (cf. relatively low numbers of identified single and multi units in Table A.10). However, the recording period of monkey L and the second recording period of monkey N were of extraordinary quality (cf. Table A.9, Table A.8, and Table A.11) with spiking activity sorted into over one hundred single units distributed over all electrodes of the array (cf. Figure 2.9).

In general, recording periods were completed when the quality of the recorded signals dropped too low. In monkey T and N this happened gradually over time, while in monkey L it happened suddenly within the last month of the recording period. At the end of the second recording period, monkey N additionally damaged the wire bundle between the connector and the array. In any case, if it was decided that a recording period was completed, for monkey T and N the retraining of the task with the other hand started. Due to the disability in the right hand of monkey L, a second recording period was not considered, and instead a surgical procedure was scheduled in which the wire bundle of the array was cut, and the connector removed. To avoid unnecessary risk of infections of an additional craniotomy, the array itself was left in place. The corresponding procedure was performed under full aseptic conditions using the same anaesthetics, and medication as used during the array implantation.

For monkey T and N this procedure was performed while implanting the second array into the other brain hemisphere.

In all monkeys, the implanted arrays were placed a few millimetres anterior to the central sulcus with the wire bundle pointing in medio-caudal direction. With the given electrode length ($400\,\mu m$) and the cortical placements of the arrays, one can assume that the arrays enabled recordings between the deep cortical layer III until the most superficial part of layer V. Based on a neutral array display, where the wire bundle point straight to the right, the arrays are rotated clock-wise around the lower left corner electrode (marked with triangle in Figure 2.2 and Figure 2.3) by 320^o and 257^o in the left hemisphere implants of monkey T (t) and N (n), as well as 218^o and 239^o in the right hemisphere implants of monkey L (l) and N (i), respectively. The stated file identifier (t), (n), and (l) indicate that these arrays were implanted for the first recording period, while the file identifier (i) indicates affiliation to the second recording period (cf. Table 2.1). In all monkeys, the aim was to position the array in the arm/hand representation of the primary motor cortex (M1) with the most anterior electrodes encroaching upon the premotor cortex (PM). With respect to the pre-central dimple and the spur of the arcuate sulcus, in monkey N the first array was located a few millimetres more medial, and the second array a few millimetres more lateral than the first arrays of monkey T and L (cf. Figure 2.2, and Figure 2.3). With respect to putative borders between MI, and dorsal as well as ventral premotor cortex (PMd and PMv), this means the following. The electrodes of the first array of monkey N are assumed to cover more or less equal parts of M1 and PMd (cf. Figure 2.2), while in monkey T and L only a few electrodes in one corner of the array are expected to cover a part of the PMd (cf. Figure 2.2, and Figure 2.3). Moreover, the anterior electrodes of the second array of monkey N are assumed to cover more likely a part of the PMv, instead of the PMd (cf. Figure 2.3). The differences of the anatomical positions of the arrays are of utmost importance during the selection of data for analysis procedures and the interpretation of corresponding results. For this reason it is not only necessary to guarantee access to the corresponding metadata (including the pictures of the implanted arrays) for all collaboration partners, but also to provide the corresponding information with each scientific publication of the experiment.

2.3 Setup

The setup was organized in three major parts, the neural recording platform, the experimental apparatus, and the behavioural control system. The components of each part and their connections are summarized in Figure 2.4 and described in more detail in the following sub-sections. While for all monkeys overall the same setup was used, the following differences need to be documented and kept in mind to be able to work with the data:

- The electrode configurations differ with each array (cf. Figure 2.8).
- As a head-stage model, Samtec was used for monkey T and L, while the Patient cable was used for monkey N (cf. in Figure 2.4, and Figure 2.7).
- The software version of Central Suite was updated after the recording periods of monkey T and L, which let to differences in the data formats of the neural raw data (ns5 in monkey T and L vs. ns6 in monkey N). Moreover, the software update made it possible in monkey N to always store the raw data (ns6) in parallel to a down-sampled and filtered version of the neuronal data in the ns2, which are called local field potential (LFP) data or signals in the following. In monkey T and L the raw data (ns5) were only saved for 13 and 10 recordings, respectively (for list of

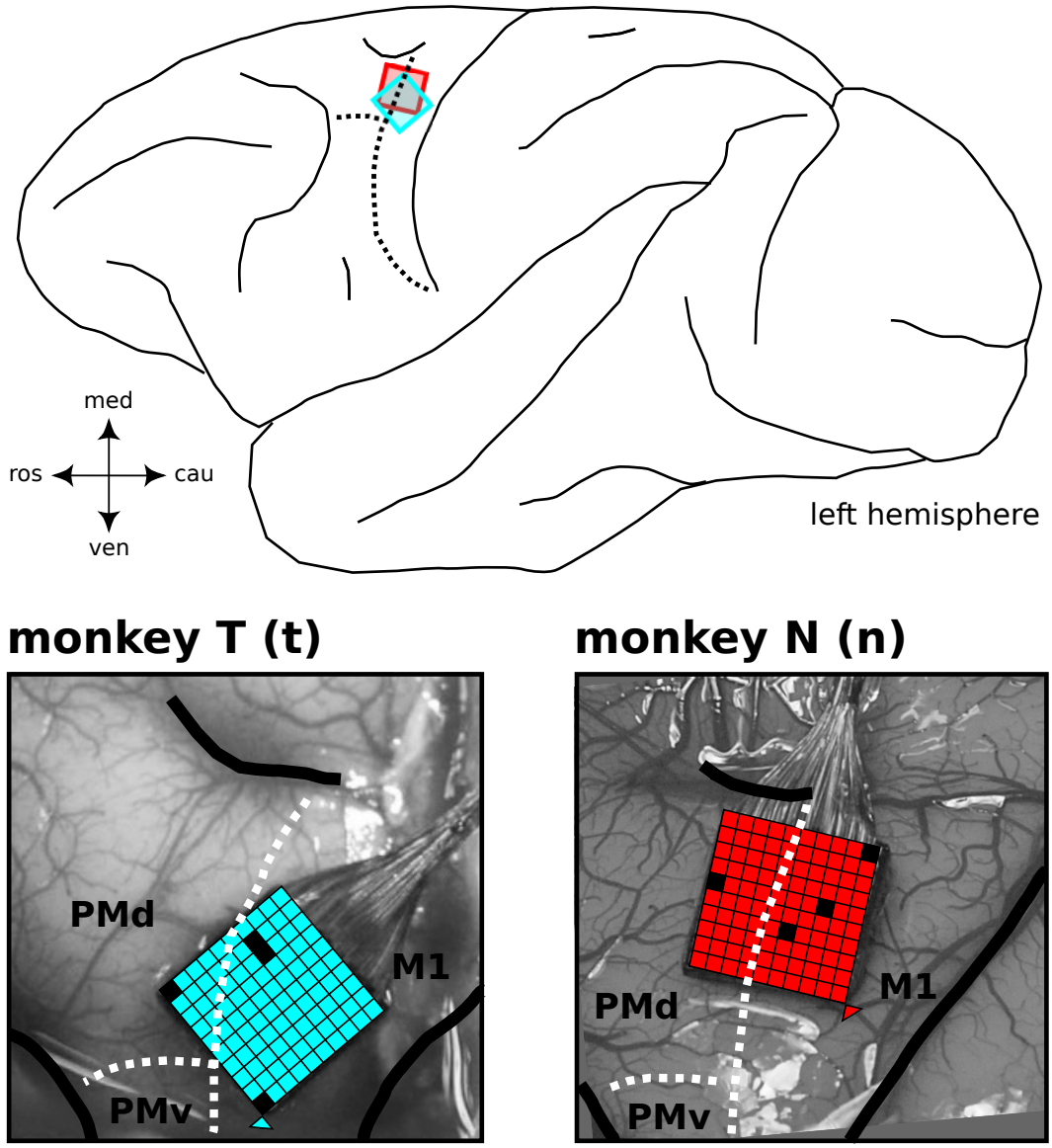

Figure 2.2 – Overview of array implants in left brain hemispheres. The upper part of the figure illustrates the array positions of monkey T (t), in cyan, and monkey N (n), in red, in a generic sketch of a left brain hemisphere of a macaque monkey. The pictures at the bottom were taken during the surgery after the array was pneumatically inserted into the cortex. They show for each monkey individually the precise implant location in respect to the putative boarders (indicated by white dashed lines in the bottom pictures and black dashed lines in the upper sketch) between primary motor cortex (M1), as well as the dorsal and ventral premotor cortex (PMd and PMv). With respect to the pre-central dimple (emphasised by a black line medial to the arrays), the spur of the arcuate sulcus (emphasised by a black line ventro-rostral to the arrays), and the central sulcus (emphasized by a black line caudal to the arrays) the array locations differ in the following way. The array of monkey N (n) is located far more medial than any other implanted array (cf. arrays of monkey T (t) as well as monkey L (l) and N (i) in Figure 2.3), with one side close and relatively parallel to the pre-central dimple. The array electrodes of monkey N (n) are assumed to cover more or less equal parts of M1 and PMd. The array of monkey T (t) is located more ventral and more close and parallel to the central sulcus. Due to the rotation, only a few array corner electrodes of monkey T (t) are assumed to cover a part of the PMd. Based on a neutral array display (wire bundle pointing to the right) the arrays are rotated clock-wise around the corner electrode which is marked in the pictures with a small triangle by 320^o and 257^o in monkey T (t) and N (n), respectively. Non-active electrodes are marked as black squares for each emphasized array in the pictures. The used file identifier (t), (n), and (l) indicate that these arrays were implanted for the first recording period, while the file identifier (i) indicates affiliation to the second recording period (cf. Table 2.1).

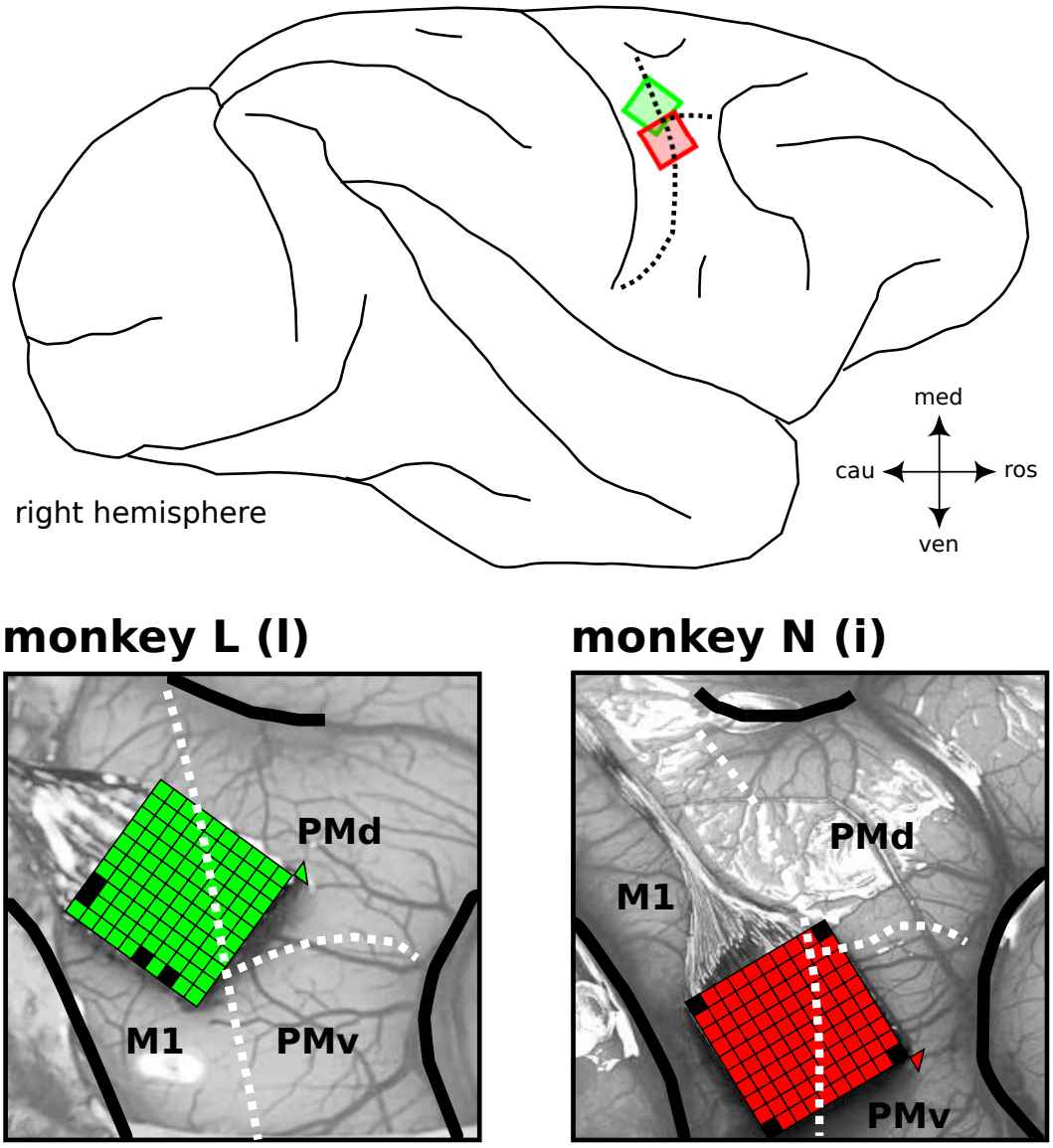

Figure 2.3 – Overview of array implants in right brain hemispheres. (modified figure 1 of the unpublished manuscript for Brochier et al., prep) The upper part of the figure illustrates the array positions of monkey L (l), in green, and monkey N (i), in red, in a generic sketch of a right brain hemisphere of a macaque monkey. The pictures at the bottom were taken during the surgery after the array was pneumatically inserted into the cortex. They show for each monkey individually the precise implant location in respect to the putative boarders (indicated by white dashed lines in the bottom pictures and black dashed lines in the upper sketch) between primary motor cortex (M1), as well as the dorsal and ventral premotor cortex (PMd and PMv). With respect to the pre-central dimple (emphasised by a black line medial to the arrays), the spur of the arcuate sulcus (emphasised by a black line ventro-rostral to the arrays), and the central sulcus (emphasized by a black line caudal to the arrays) the array locations differ in the following way. The array of monkey N (i) is located far more ventral than any other implanted array (cf. arrays of monkey L (l) as well as monkey T (t) and N (n) in Figure 2.2), so that its corner electrodes are assumed to cover more likely a part of the PMv instead of PMd. The array of monkey L (l) is positioned more similar to the location of the array of monkey T (t) (cf. Figure 2.2), with a few corner electrodes that are assumed to cover a part of the PMd. Based on a neutral array display (wire bundle pointing to the right) the arrays are rotated clock-wise around the corner electrode which is marked in the pictures with a small triangle by 218^o and 239^o in monkey L (l) and N (i), respectively. Non-active electrodes are marked as black squares for each emphasized array in the pictures. The used file identifier (t), (n), and (l) indicate that these arrays were implanted for the first recording period, while the file identifier (i) indicates affiliation to the second recording period (cf. Table 2.1).

datasets cf. filename entries in Table 2.6), and the ns2-files saved in parallel only contain the behavioural signals from the target object manipulation and not the LFP signals additionally, as in monkey N.

- The filter frequencies and the waveform window size for the online spike extraction in Central Suite are not identical between the monkeys (cf. Figure 2.7). For monkey T and L the analogue signals were filtered for the online spike extraction with a first order high-pass filter with cut-off frequency of $500Hz$, while for monkey N a cut-off frequency of $250Hz$ was used. The window size of the extracted waveforms for each detected spike was set to 48 samples ($1.6ms$) in monkey T and L, and reduced to 39 samples ($1.3ms$) in monkey N.

- Implemented in LabView, the program to control and monitor the task and behaviour was updated after the recording periods of monkey T and L. This led to differences in the binary coding of the digital events stored in the nev-file (cf. in Table 2.4).

The differences are described in more detail below, and are marked in cyan, green and red in the referenced figures for monkey T, L and N, respectively.

2.3.1 Experimental apparatus

The experimental apparatus was composed of a table switch, a target object, a visual cue, and a reward system. On each recording day, the monkey was seated in a custom-made primate chair and placed in front of that apparatus (cf. Figure 2.1). The non-working arm of the monkey was loosely restrained in a semi-flexed position. To control the home position of the working hand between the reach-to-grasp movements, the **table switch**, which was installed close to the monkey at waist level, $5\,cm$ lateral to the mid-line, needed to be pressed down (cf. Figure 2.1).

The **target object** was a stainless steel rectangular cuboid ($40mmx16mmx10mm$) rotated $45degrees$ around the vertical axis and pointing towards the monkey (cf. Figure 2.1). It was located $13\,cm$ away from the table switch at $14\,cm$ height. The posterior end of the object was attached via a low-friction horizontal shuttle to a counterweight hidden inside the apparatus, which was used to define the object load. The object load was set to 1 of 2 possible values to define the force type (LF and HF) needed for pulling the object in each trial of the standard task (cf. Section 2.1.1) by deactivating and activating an electromagnetic weight resting below the counterweight inside the apparatus (cf. Figure 2.5). When activated, it attached to the counterweight and increased overall weight from usually $100\,g$ to $200\,g$, which corresponds roughly to a pulling force of$1\,N$ and $2\,N$ for LF and HF, respectively. However, on three recording days of monkey L (2011-03-30/31, 2011-04-01) the HF object load was gradually increased to $300g$ and $350g$ in subsequent recordings by adding weight to the electromagnet and/or the counterweight, to test if the increase in LF/HF differences also emphasizes differences in the recorded neuronal activity between the two force types. The measured effect was indeed minimal, so that the change in settings was abandoned again. In connection with the electromagnet, two further peculiarities need to be documented. First, at the beginning of the very first recording period with monkey T, the activation and deactivation of the magnet to set the object load happened during a trial before the first cue (between WS-ON and CUE-ON, see Section 2.3.2 and *panel a* Figure 2.6). However, it was realized that this state switch of the magnet led to a sudden jump in the base line of the analogue signals of the object-attached sensors which monitored the monkey's reach-to-grasp behaviour (see below). To gain a stable base line during the trials for these signals for all upcoming recordings after April 23, 2010, the magnet state switch was timed to the trial start. The other peculiarity that is important to know when analysing the data, is that the electromagnet failed in

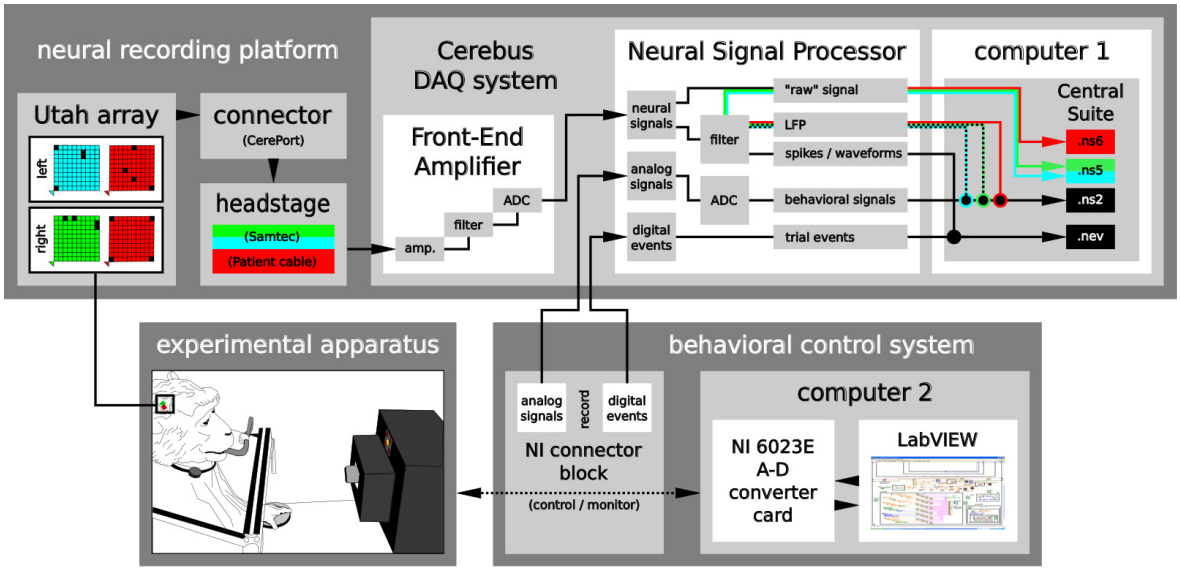

Figure 2.4 – Overview of the setup components. (modified figure 2 of the unpublished manuscript for Brochier et al., prep) The setup of the R2G experiment is composed of three major parts: the neural recording platform, the experimental apparatus, and the behavioural control system. The neural recording platform (top) was composed of the implanted Utah-Array (of left or right brain hemisphere) with its corresponding connector (CerePort), a headstage (Samtec or Patient cable), and the Cerebus data acquisition (DAQ) system (i.e. the Front-End Amplifier, Neural Signal Processor (NSP), and the Cerebus control software, Central Suite, installed on the setup computer 1). The experimental apparatus (bottom left) consisted of the physical devices which the monkeys had to interact with (i.e., the visual cue system, the target object, the table switch, and the reward system). The behavioural control system (bottom right) was build from hard and software of National Instruments (NI, National Instruments Corporation, Austin, Texas, USA). It was composed of a NI connector block which was linked via a NI 6023E A-D converter card to the setup computer 2 on which the NI system design software, LabView, was running. To record the behavioural data the behavioural control system was interlinked with the neuronal recording platform via the NSP and the NI connector block. All three parts are separately described in more detail in main text, as well as Figure 2.6 and Figure 2.7. Setup differences between the monkeys are indicated in cyan, green, and red, for monkey T, L, and N, respectively.

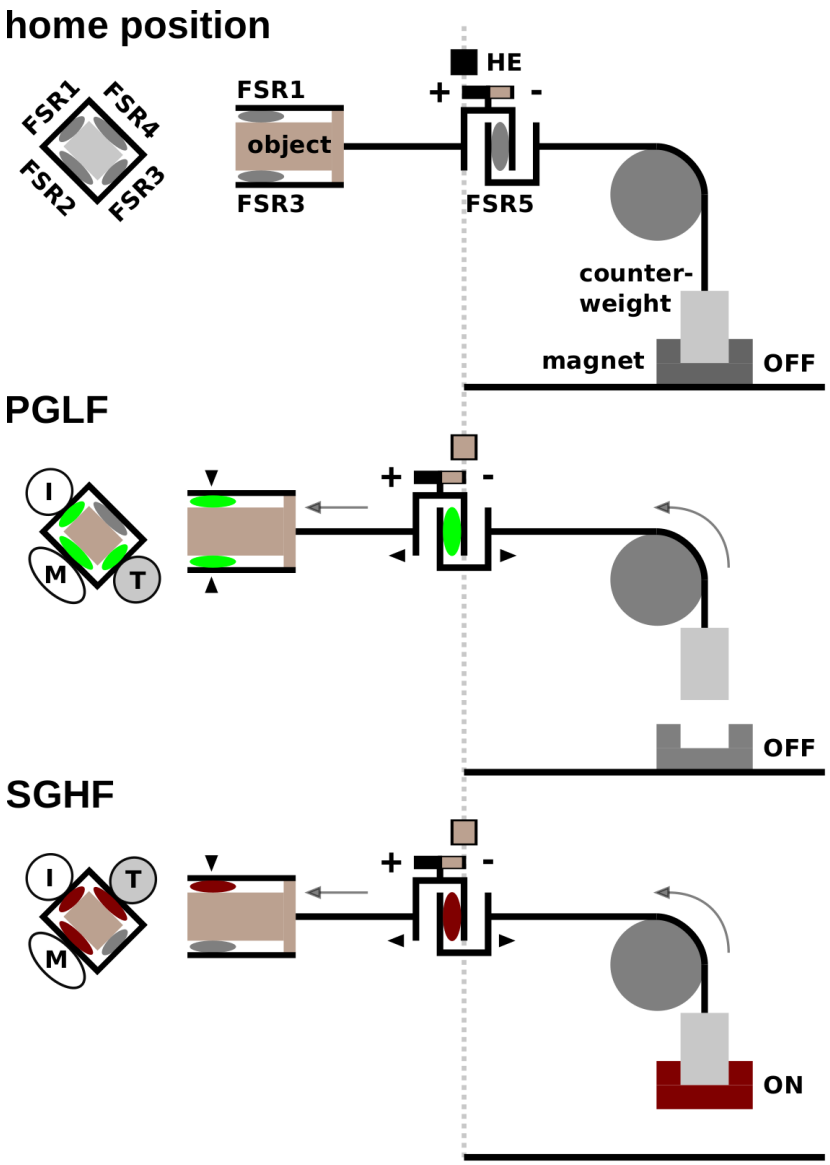

Figure 2.5 – Overview of the target object system. Each row of the figure shows a sketch of the state of the target object system (front and side view) while the monkey is resting in its home position (top row), or performing a reach-to-grasp movement using a precision grip for a low object load (PGLF, middle row), or a side grip for a high object load (SGHF, bottom row). The elements of the target object system, namely the object, the five force sensitive resistance sensors (FSR1-FSR5), the hall effect sensor (HE), as well as the object's counterweight and corresponding electromagnet (magnet), are all labelled in the home position sketch. Mark that the HE sensor is fixed inside the object system above a small magnet which is attached to the FSR5 sensor. When the object is pulled out by the monkey (indicated by long grey arrows), this magnet is displaced modulating the voltage signal of the not moving HE sensor (indicated by colour change of the HE sensor in the PGLF and SGHF sketch). The FSR5 sensor signal is modulated by the pull force and the opposed gravitational force of the object load (indicated by black arrow heads around the FSR5 installation in PGLF and SGHF sketch). The object load is defined by the counterweight and the weight of the electromagnet, if it was activated and is attached to the counterweight (ON vs. OFF for HF vs. LF trials). The remaining four FSR sensors are located at the object's top, left, bottom, and right side (FSR1, FSR2, FSR3, and FSR4, respectively) and measure the grip force of the monkey's index finger (I), middle finger (M) or thumb (T) for a PG, or a SG (indicated by the fingers and the coloured sensors in the front view of the object, and by black arrow heads and coloured sensors in its side view in the PGLF and SGHF sketch). See also Table 2.3 for additional information on the object sensors.

two recordings during the recording period of monkey L (l101111-001 and l110209-004). As already mentioned, the object was equipped with six sensors which monitored the monkey's reach-to-grasp behaviour (for an overview see Figure 2.5). Four force sensitive resistance (FSR) sensors on the object surface provided continuous measurement of the grip forces applied on the object sides by the index and middle finger, as well as the thumb. The different activation pattern of these four FSR sensors, in particular the different placement of the thumb (cf. finger positions for different grip types in Figure 2.1), was used to detect online if the grip type was performed correctly. An additional FSR sensor installed between the object and its counterweight. This FSR sensor was used to measure the horizontally applied force needed to oppose the corresponding object load. Due to the low, but still existing friction of the object moving inside the horizontal shuttle, the measured force signal of this sensor is not perfectly proportional to the horizontal force needed to lift the opposed object load, but sufficient to distinguish between LF and HF settings (cf. example in bottom right panel of Figure A.1 and Figure A.2). The displacement of the object along the horizontal shuttle over a maximal distance of $15\,mm$ was measured by a hall-effect (HE) sensor. All sensors of the object are summarized in Table 2.3 and Figure 2.5.

object sensors	used label	attached to	activated by	used to identify	ns2 channel ID
FSR 1	GF pr1	top	index finger	PG	137
FSR 2	GF side1	left	middle finger	SG	138
FSR 3	GF pr2	bottom	thumb	PG	139
FSR 4	GF side2	right	thumb	SG	140
FSR 5	LoadForce	spring	pull	load force	141
HE	Displ	shuttle	displacement	displacement	143

Table 2.3 – Overview of the attached object sensors monitoring the monkey's grasp behaviour. The object has 5 force sensitive resistance (FSR) sensors and one hall effect (HE) sensor. The first four FSR sensors are located at the object's top, left, bottom, and right side and measured on activation by touch of index finger, middle finger or thumb, the grip force (GF) of precision (pr) or side grip (PG and SG, respectively). The fifth FSR sensor is installed between the object and its counterweight and measured therefore the load force when the object was pulled by the monkey (pull). The HE sensor was attached to the horizontal low-friction shuttle along which the object could be moved, and measured the actual object displacement (Displ). See also Figure 2.5 for more information on the object sensors.

The **visual cue system** was composed of a $10\,mm\,x\,10\,mm$ square equipped with five LEDs and located just above the target object. It was used to instruct the monkey about the requested behaviour. While the central yellow LED was used to warn the monkey that a trial had started, the four red corner LEDs were used to code separately the grip and the force type for the requested trial type of each trial. In this context the illumination of the two left, the two right, the two bottom, or the two top LEDs coded for SG, PG, LF, or HF, respectively (see Figure 2.1 for illustration).

The **reward system** consisted of a bucket filled with apple sauce which was equipped with a feeding tube and a pump to deliver the reward (few drops of the apple sauce) to the monkey (cf. sketches in Figure 2.1) on demand.

2.3.2 Behavioural control system

The core of the behavioural control system is a home-brewed program in LabView that controls the digital event sequence and the requested behaviour of each trial in a recording. A digital event reflects hereby the activation or deactivation of a physical device of the experimental apparatus. In this context, the LabView program is responsible to activate and deactivate the LEDs of the visual cue system, the reward pump, and the electromagnet. The latter is not controlled by a digital event, but

by an analogue square signal that switches the magnet on or off. To control the requested behaviour, the LabView program monitors the monkey's manipulation of the table switch and the target object. Table switch as well as all sensors of the target object produce continuous analogue signals that are digitized by the NI converter card and fed into the LabView program of the setup computer (see Figure 2.4 computer 2). The square signal of the table switch is then reinterpreted online as a digital activation or deactivation event. Figure 2.6 displays in *panel b* a schematic diagram of how the physical devices of the experimental apparatus are connected to the setup computer (computer 2) where the LabView program controls the sequence of digital trial events, and, in parallel, monitors the monkey's reach-to-grasp behaviour via the analogue signals of the object sensors, both sketched in *panel a*.

In the following, I will describe a typical execution of the LabView program during a recording in more detail.

Before a recording started, two settings of the program needed to be configured to determine the overall task and task condition (cf. Section 2.1). For this, first, the possible trial types, and second, their possible sequence needed to be fixed (cf. trial type generator in b of Figure 2.6). As already summarized in Section 2.1 and Table 2.2, the possible trial types were set according to the chosen task condition. The alternation of the chosen trial types in sequence was then adjusted to be random or repeating in blocks of a fixed trial count, with the possibility to set the sequence order for grip and force types independently. For example, for a recording of a standard task of condition 1 (cf. Section 2.1.1 and Table 2.2), the trial type sequence could be generated with one of the four following settings: (i) both, grip and force, types alternate randomly, (ii) the grip types alternate randomly, while the force types alternate in blocks of, e.g., 10 trials, (iii) the grip types alternate in blocks of, e.g., 10 trials, while the force types alternate randomly, and (iv) both grip and force types alternate in blocks of, e.g., 10 trials. Once all settings of the overall task were defined, the LabView program was started to repetitively run and control the event sequence and behaviour for each trial during the recording. Each single trial was run and controlled as follows:

The LabView program only started a trial when the monkey deactivated the table switch by pressing and holding it down (cf. home position in Figure 2.1). This does not require much muscle activity, but simply the weight of the monkey's hand on top of the smooth-running switch. If the table switch was deactivated, the LabView program internally initiated a trial with a short time delay (cf. TS-ON in Figure 2.6). In parallel, the program randomly picked one of the possible trial types (e.g., PG-HF) and activated or deactivated the electromagnet accordingly to fit the chosen load force of the object (e.g., activated for HF). To inform (or warn) the monkey that a new trial has started, the central LED (LED c) was illuminated $400\,ms$ after the trial was initiated by the program (cf. WS-ON in Figure 2.6). $400\,ms$ after WS-ON, the grip or force type was revealed to the monkey by illuminating the corresponding corner LEDs of the chosen trial type (cf. CUE-ON, e.g., top and bottom right LED (LED tr and br) for PG-ON in Figure 2.6). The LEDs of this first cue were turned off again after $300\,ms$ (cf. CUE-OFF in Figure 2.6). The CUE-OFF was followed by a $1000\,ms$ preparatory delay at the end of which the monkey was informed about the upcoming force or grip type, again by illuminating the corresponding corner LEDs of the chosen trial type (cf. GO-ON, e.g., top left and right LED (LED tl and tr) for HF-ON in Figure 2.6). This second cue also functioned as a GO signal for the monkey to initiate the movement, which was registered when the active hand of the monkey released the table switch (cf. SR-ON in Figure 2.6) after a variable reaction time (RT). The execution of the movement was composed of reaching, grasping, pulling and holding the object in the position window for $500\,ms$. The LabView program controlled the movement execution online by checking the used grip type, the object displacement and the hold time. For checking the grip type, the grasp of the object was registered by small deflections of the FSR surface sensor signals caused by the monkey's

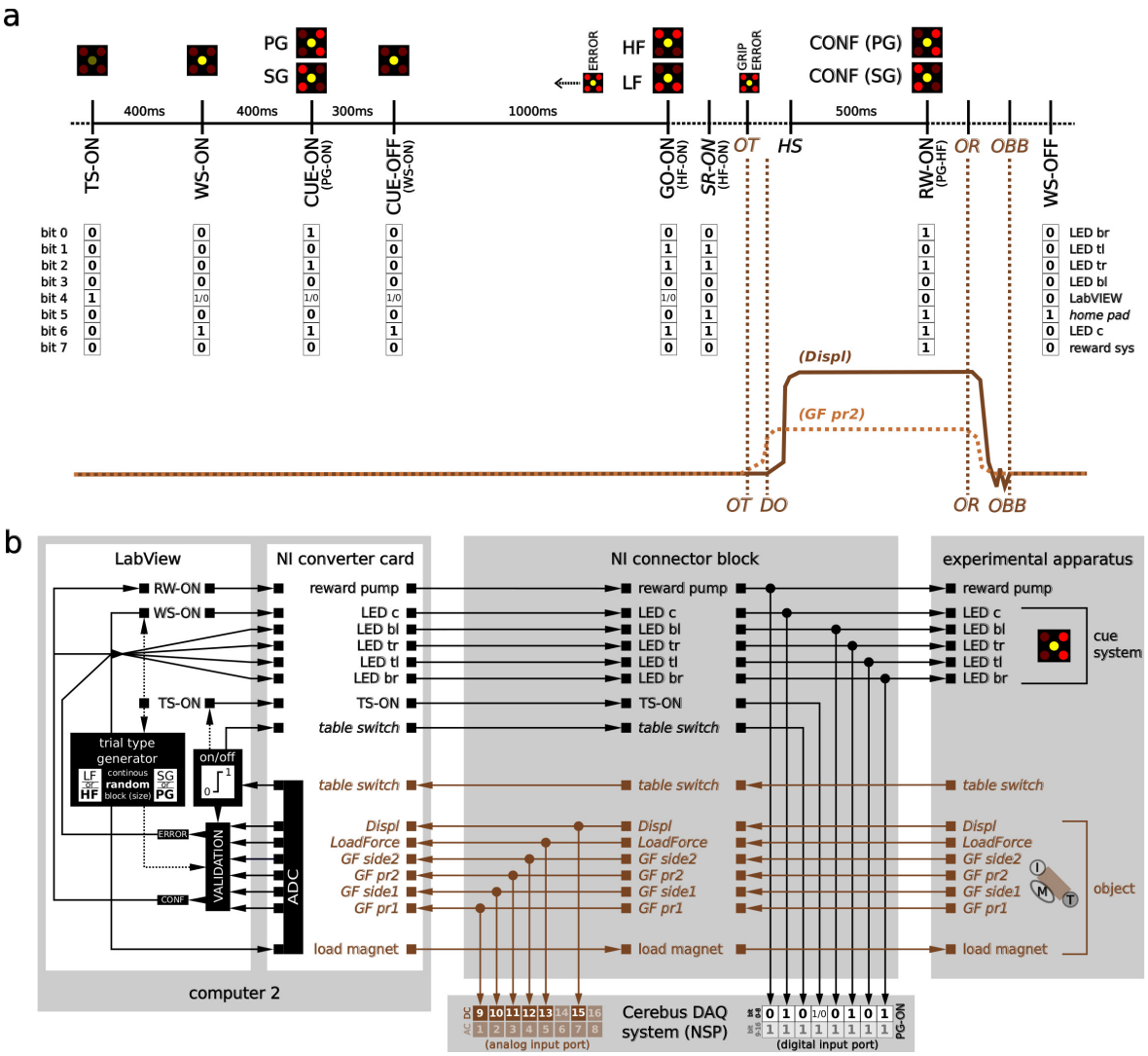

Figure 2.6 – Overview of the behavioural control system. (modified figure 3 of the unpublished manuscript for Brochier et al., prep) ***a)*** Trial scheme of a typical event sequence with the respective visual cues (different illumination combinations of the 5 LEDs illustrated on top) that were shown to the monkey at the respective time points. The events are marked along the time axis (see main text for abbreviations). Events with black font mark digitally recorded events, whereas events with brown font indicate events which were extracted offline from baseline deviations of the analogue signals of the object's sensors (cf. Section 2.4.2). Additionally, events which were generated by the monkey are emphasized in italic fonts, while all upright written events are produced by LabView. The 8-bit binary code for each digital event sent from the LabView program to the NSP with the respective time is shown below the time axis of the trial scheme. Example traces for the analogue signals of the HE sensor (Displ; dark solid line) and one of the 4 FSR sensors located at the object's surface (GF pr2, light dotted line) used to monitor online the monkeys behaviour are shown at the bottom the trial scheme. The indicated behavioural events (OT, DO, OR and OBB) were extracted in an offline executed pre-processing step (cf. Section 2.4.2). ***b)*** Outline of the devices and their wiring controlling the behaviour. All analogue signal streams are coloured in brown, whereas all digital signal streams are coloured in black. As reminder, an inset of the position of the index finger (I), the thumb (T), and the middle finger (M) on the object (brown cube) and the illumination of the visual cue system for a PG are shown within the experimental apparatus schema.

fingers. A FSR sensor was registered as activated if the deflection surpassed a predefined threshold. The pattern of activated FSR sensors was then used by the LabView program to control if the monkey performed the requested grip type. This meant, in particular, to check for SG and PG, if the FSR sensor on the right (GF side2), or on the bottom (GF pr2) of the object was activated by the monkey's thumb, respectively (cf. Figure 2.1 and Table 2.3). The other two sensors that measured force from the index and middle fingers for the two grip types (GF side1, and GF pr1) were not controlled online. Nonetheless, all sensors were used in an offline preprocessing step to determine the exact time points of the first object touch (OT), the object displacement onset (DO), the object release (OR) and when the object reached back its baseline position (OBB). This preprocessing step will be explained in more detail later on (cf. Section 2.4.2). If the correct grip was detected, the grip cue was illuminated again as a positive feedback (cf. CONF in Figure 2.6). To check the object displacement, the LabView program measured if the deflection of the HE sensor signal of the object was within the two defined position thresholds ($4\,mm$ and$14\,mm$). The time point at which the displacement signal surpassed the lower threshold was used by the LabView program to define the start of the holding period (cf. HS in Figure 2.6). If the object remained within the position window for $500\,ms$ after HS was set, LabView activated the reward pump which provided the monkey with a few drops of apple sauce as reward for a successful trial. The time point of HS is not captured as separate event, but later on assumed to be $500\,ms$ before RW-ON. The monkey was allowed to release the object at its own pace as soon as it received the reward. To indicate the monkey that a running trial ended, LabView turned off all LEDs ,which happened for successful trials often in parallel to the deactivation of the reward pump (cf. WS-OFF in Figure 2.6). A new trial sequence was started by LabView (TS-ON) as soon as the monkey returned into the home position (new deactivation of the table switch).

An abort of the described trial sequence by LabView (error trial) was triggered by the following three scenarios: (i) the monkey released the table switch before the GO cue, (ii) the wrong grip type was registered, and (iii) the object was not pulled and held long enough in the position window. In case one of these scenarios were registered by LabView, the trial was aborted and all LEDs were turned off to indicate the monkey that the trial ended (WS-OFF). In addition, for monkey T and L, the LabView program provided a negative feedback when aborting a trial by flickering all LEDs three times (cf. ERROR or GRIP ERROR in Figure 2.6).

For storing the trial event sequence and the analogue signals of the object sensors along with the neuronal data registered via the neural recording platform, the behavioural control system was connected to the NSP of the Cerebus DAQ system (see *panel b* of Figure 2.6). For this, the analogue signals of the sensors of the target object were copied from the NI connector block to the analogue input port of the Cerebus System NSP via DC coupled BNC cables and connectors. In the NSP, the behavioural analogue signals were digitized with a $16-bit$ resolution at $0.15\,mV/bit$ and a sampling rate of $1\,kHz$ and saved in the ns2-file (cf. Section 2.3.3 and Figure 2.7). All digital or digitized events that register the activation and deactivation of the table switch, the LEDs of the cue system, and the reward pump, as well as the internally generated digital trial start event (TS-ON) are coded into a $8-bit$ binary signal (cf. Figure 2.6) and transferred via the NI connector block to a $16-bit\ DB-37$ input port of the NSP, where they occupy the first $8\,digits$ (remaining digits are set to 1). In the NSP the now $16-bit$ binary signal of each event is stored in its decimal representation and corresponding time point in the nev-file (cf. Section 2.3.3 and Table 2.4).

2.3.3 Neuronal recording platform

The recording of the neural signals was performed using a neural recording platform with components produced by Blackrock Microsystems[2] (Salt Lake City, UT, USA). The platform consisted of the multi-electrode Utah-Array, a headstage, and a Cerebus data acquisition (DAQ) system. The latter is composed of a Front-End Amplifier, and a real-time Neural Signal Processor (NSP) and the control software, Central Suite (version 4.15.0 and 6.03.01 for L and N, respectively), running on Windows XP for monkey T and L, and Windows 7 for N on the setup computer 1 (see Fig. Figure 2.7). The Cerebus DAQ system was also connected to the behavioural control system via the NI connector block to save the analogue behavioural data and digital trial event signals that were described in the previous section, in parallel with the neural signals. All data were transmitted from the NSP via an Ethernet cable to be saved first locally on the setup computer 1. All data files (nev, ns2 and or ns5/6, as well as a Cerebus configuration file) were saved on disk and backed-up on a data server at the end of each recording day. In the following, I will describe the function of the different components of the neural recording platform in more detail.

Each Utah-Array was linked via a CerePort connector to an analogue Blackrock headstage with unity gain to reduce the environmental noise. Overall, the reduction of the noise was better with the Patient Cable headstage which was used in monkey N, than with the Samtec headstage used in monkey T and L.

The used Utah-Arrays as well as their anatomical implant location in each monkey was described previously (see Section 2.2.2, Figure 2.2, and Figure 2.3). In Figure 2.8, I give a more detailed documentation of the electrode configurations of each array.

In the Front-End Amplifier, each of the 96 neural signals of the Utah-Array electrodes was differentially amplified with respect to the reference of its corresponding input port (gain 5000) and filtered with a 1st-order $0.3\,Hz$ high pass filter (full-bandwidth mode) and a 3rd-order $7.5\,kHz$ Butterworth low pass filter. After that, the band-pass filtered neuronal signals were digitized with a 16-bit resolution at $0.25\,V/bit$ and a sampling rate of $30\,kHz$, in the following called "raw signal". The digitized signals were converted into a single multiplexed optical output and transmitted via a fibre-optic data link to the neural signal processor (NSP). The raw signal ($0.3\,Hz - 7.5\,kHz$) was saved for a few recordings in an ns5-file for monkey T and L and for nearly all recordings in an ns6-file for monkey N. The file format depended on the firmware and software version of the Cerebus DAQ system (cf. Central Suite versions in Figure 2.7). In addition to the neuronal signals, the NSP received, via its analogue input port, the analogue signals of the object sensors recorded by the behavioural control system. These behavioural signals were digitized and saved with a sampling rate of $1\,kHz$ in an ns2-file. For monkey N, the ns2-file also always contained a filtered and down-sampled version of the raw signals, in the following called "LFP data". For monkey T and L, the LFP data were usually saved instead and never in parallel to the raw signals. If saved, the LFP data were extracted for all monkeys by creating a copy of the raw data which was then online digitally band-pass filtered between $0.3\,Hz$ and $250\,Hz$ (Butterworth, 1st and 4th order, respectively), and down-sampled to $1\,kHz$ within the NSP.

The NSP also performed an online spike waveform detection and classification controlled via Central Suite. The sorted spikes were used for a first online inspection of the data as well as for selecting and saving the spike waveforms for offline sorting. For this purpose, the neuronal raw signals were online high-pass filtered above $500\,Hz$ for monkey T and L and $250\,Hz$ for monkey N (causal, Butterworth, 1st order), and the potential spikes were detected by threshold crossing (manually set). Each time the high-pass filtered signal passed the threshold, a snippet of $1.6\,ms$ ($48\,samples$) for monkey

[2] www.blackrockmicro.com

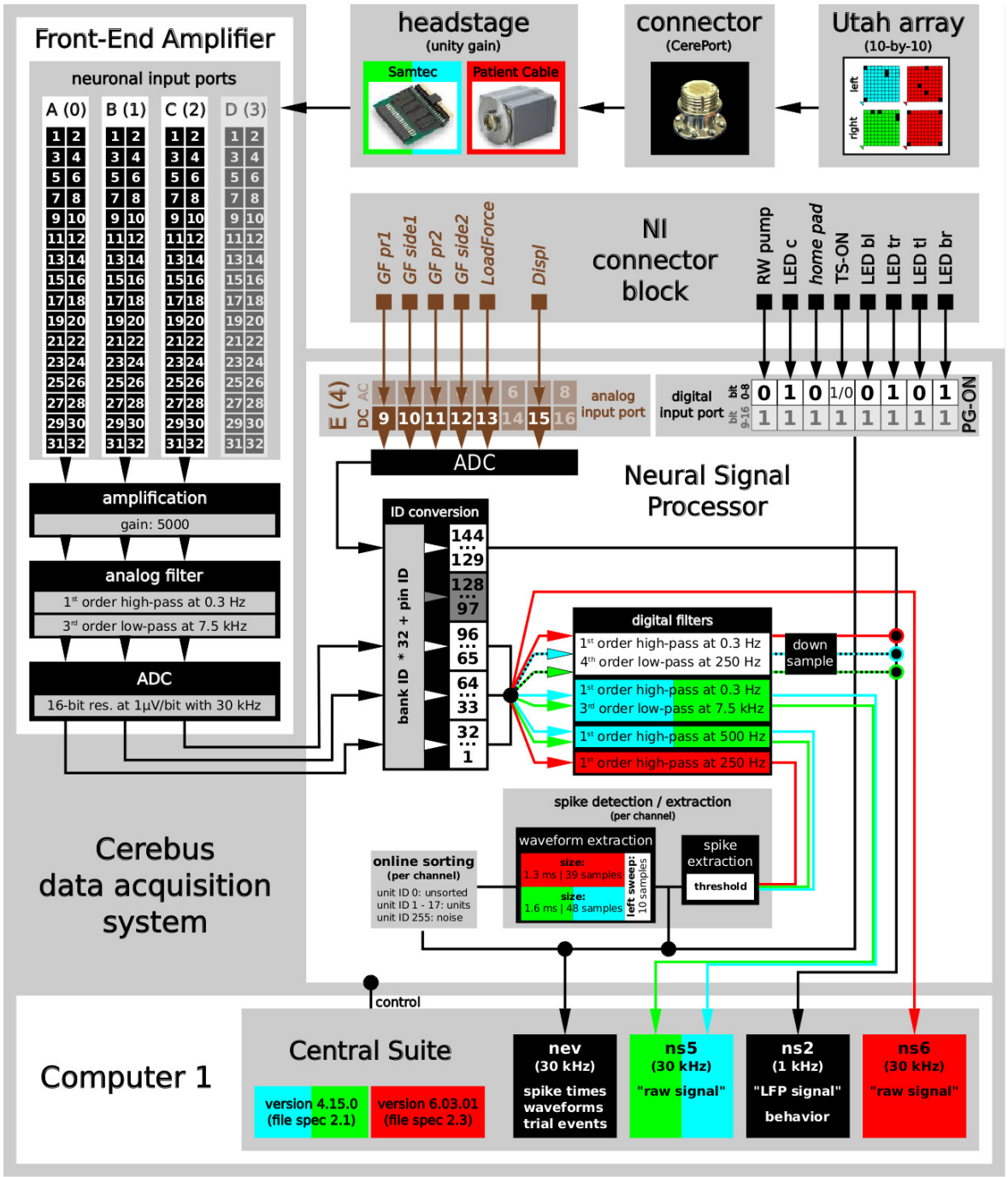

Figure 2.7 – Overview of the neuronal recording platform. (modified figure 4 of the unpublished manuscript for Brochier et al., prep) Data were recorded using a Utah-Array, which was linked via its connector (CerePort) to a headstage (Samtec or Patient Cable) with a unity gain. From there the neural signals were transferred to the Cerebus Front-End Amplifier, where the neural signals where amplified, filtered and digitized. The digitized signals were converted into a single multiplex optical output and sent via a fibre-optic data link to the Neural Signal Processor (NSP) which is controlled by the Cerebus control software, Central Suite (version 4.15.0 or 6.03.01). Within the NSP the time points and waveforms of potential spikes were extracted online from a correspondingly processed copy of the neural signals and saved in an nev-file. The continuous raw signals (sampled at $30\,kHz$) were stored in an ns5 or ns6-file. A filtered and down-sampled version of the neural signals ($0.3\,Hz - 250\,Hz$ sampled at $1\,kHz$) was saved in an ns2-file, either parallel to an ns6 (indicated by solid lines) or instead of an ns5-file (indicated by dashed lines). Simultaneously to the neural signals, the NSP also received the digital trial event produced by the LabView program, and the analogue signals of the object sensors via the NI connector block of the behavioural control system (cf. Figure 2.6). While the digital trial events were saved along with the extracted potential spikes in the nev-file, the analogue signals of the sensors were digitized and saved in an ns2-file. Components and settings specific to monkey T, L and N are indicated by cyan, green, and red, respectively.

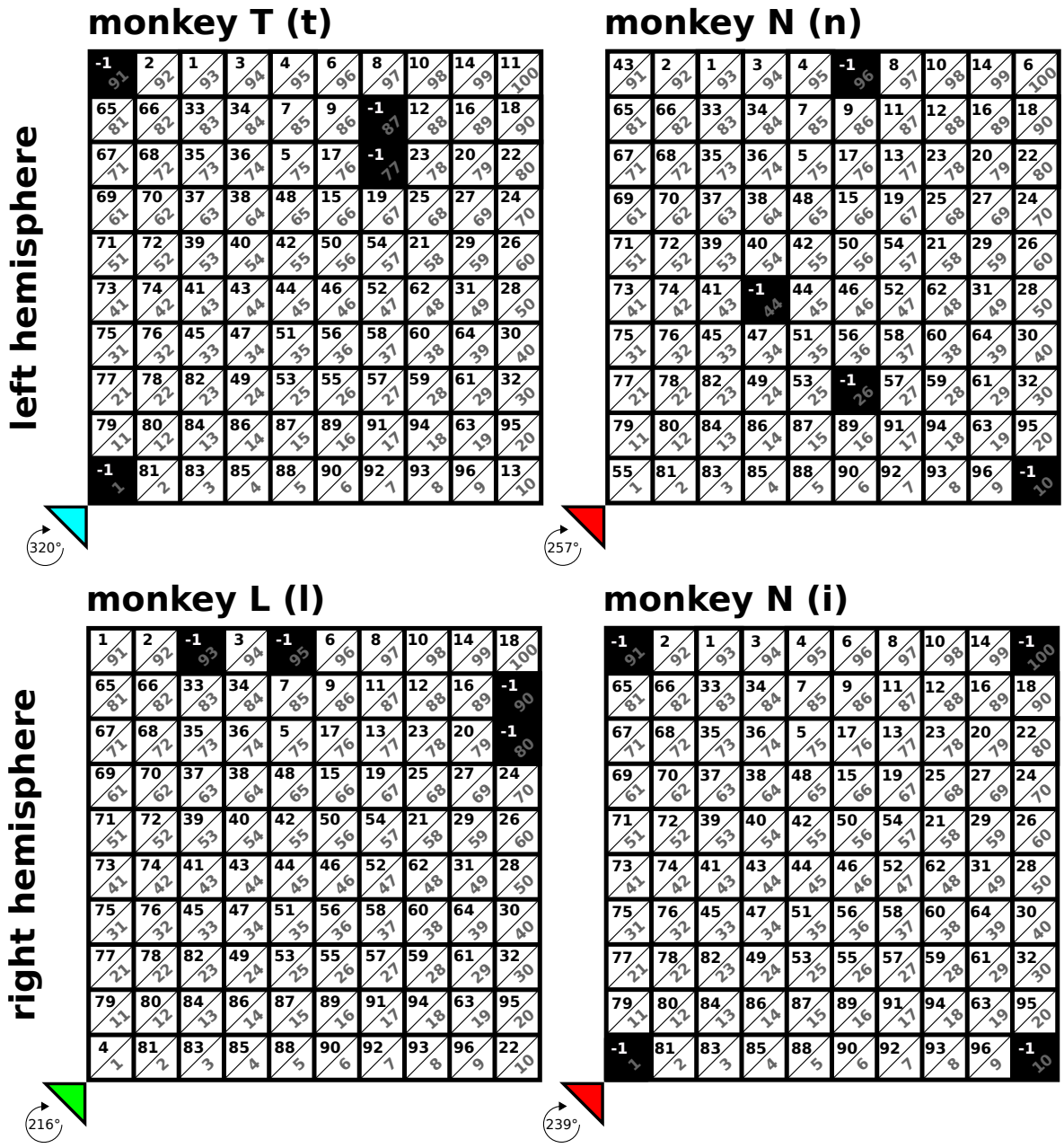

Figure 2.8 – Overview of the electrode configuration of the implanted Utah-Arrays. (partially based on figure 1 of the unpublished manuscript for Brochier et al., prep) The identification number (ID) of each array electrode (cf. upright, black numbers) is calculated by Central Suite within the Neural Signal Processor (NSP) and depends on how the electrodes are connected with the input ports of the Front-End Amplifier. The Front-End Amplifier has in total four neuronal input ports (cf. Figure 2.7). Each port consists of a male header with 34 pins of which 32 are reserved for neuronal signals. The remaining two are used as reference and ground, respectively. With 96 active electrodes, the Utah-Arrays only occupied input ports A-C (IDs 0, 1, 2). For each electrode, Central Suite multiplies the corresponding port ID with 32 and adds the ID of the actual pin the electrode is connected to (cf. ID conversion in Figure 2.7). Although the fabrication process of Utah-Arrays in automatized, the electrode wiring of the Utah-Array cannot be ordered to the input ports and pins of the Front-End Amplifier which leads to a spatially unordered electrode ID configuration. In addition, the grid position of the four unconnected electrodes can be modified. As default, the four corner electrodes are not connected, as it is the case for monkey N (i). If in the fabrication process a corner electrode was registered to be of significantly higher quality than another electrode of the grid, the corner electrode was connected instead, as one can see for monkey T (t), L (l), and N (n). To facilitate the comparison of results between arrays, uniform IDs were assigned based on a neutral array display where the wire bundle to the connector is pointing to the right. Starting from the lower left corner electrode (marked with a triangle) these uniform, connector aligned IDs (caIDs) increased linearly from bottom left to top right, line by line (cf. titled, grey numbers). Rotating the arrays clock-wise around the marked corner electrode by the given degrees provides the actual implant orientation in the stated brain hemisphere (cf. Figure 2.2 and Figure 2.3).

T and L and 1.3 *ms* (38 *samples*) for monkey N was cut and saved as potential spike waveform. The snippet was cut with 10 sample points before threshold crossing and 38 or 28 points after for monkey L or N, respectively. To get an overview of the general quality of the spiking activity in the data during the recordings, the extracted waveforms were displayed in the online classification window provided by Central Suite and manually pre-sorted using the Hoops spike sorting method. The Hoop spike sorting is derived from a simple, in Central Suite implemented, decision rule, which, for each channel, assigns the extracted waveforms to up to five units, if the corresponding signal passes through one of five possible amplitude windows at a manually chosen point in time of the waveform. These amplitude windows are termed hoops and can be further modified by manually changing their corresponding lower and upper signal thresholds. Waveforms that were online identified as potential units via this Hoops spike sorting method were labelled with unit IDs from 1 to 5. Unsorted waveforms were labelled with ID 0. Note here that Central Suite can assign unit IDs from 0 to 16 (potential neurons) and a noise "unit" with ID 255 via other online spike sorting methods. At the end of a recording, all extracted waveforms were saved together with their respective time stamps and their unit ID resulting from the online pre-sorting in a nev-file. The thresholds (one for each channel) for the spike waveform detection were not modified during a recording day and saved in a Cerebus configuration file (ccf) for each recording along with all other settings and configurations of Central Suite (e.g., the selected filter settings). Without requiring the presence of the Cerebus DAQ system, the ccf can be used offline with a software, called CentralPlay, to visually inspect the data with the used settings. Nevertheless, due to the high number of electrodes, the online spike-sorting was only moderately reliable. For this reason, the spiking activity was re-sorted offline on each channel using a Plexon Offline Spike Sorter (Plexon Inc., Dallas, Texas, USA, version 3.3, for details see Section 2.4.3). The results of the offline sorting were saved in a copy of the original nev-file with an updated file name.

2.4 Technical validations

After the recordings, a number of pre-processing steps (*pre* in the sense of before the actual upcoming data analysis, but being the post-processing after the recording) were performed. This includes (i) the translation of digital events from their binary codes set by the Cerebus DAQ system to a human-readable format and putting them in context of the expected executed trial event sequence, (ii) the offline detection of behavioural trial events and object load force from the analogue signals recorded by the sensors of the target object, and (iii) the offline spike sorting.

In addition to the above described preprocessing steps that needed to be performed to gain more content from the primary data, some technical validations of the data also had to be conducted. These technical validations included (i) the correction of the irregular alignment data files of the Cerebus DAQ system, (ii) the correction of data gaps caused by an hardware failure of the Cerebus DAQ system in monkey T and L, and (iii) a general quality assessment of the data. In order to validate the quality of the recording, a series of algorithms were applied to the data. For the LFP signals, the quality of the recording was assessed per electrode and per trial by evaluating the variance of the corresponding signal in multiple frequency bands. In a separate validation, the quality of the offline sorted single units (Section 2.4.3) was determined by a signal-to-noise measure. In addition, noise artefacts occurring simultaneously in the recorded spiking activity were detected and marked. In the following subsections, I explain in more detail the execution of the preprocessing steps and the technical validation of the data.

decimal code	(8-bit) binary code								interpretation		
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0			
65296	0	0	0	1	0	0	0	0	**TS-ON**		(T,L,N)
65280	0	0	0	0	0	0	0	0	**TS-OFF/STOP**		(T,L)
65344	0	1	0	0	0	0	0	0	**WS-ON/CUE-OFF**		(T,L)
65360	0	1	0	1	0	0	0	0			(N)
65349	0	1	0	0	0	1	0	1	**PG-ON**	(CUE-ON/GO-ON)	(T,L)
65365	0	1	0	1	0	1	0	1			(N)
65354	0	1	0	0	1	0	1	0	**SG-ON**		(T,L)
65370	0	1	0	1	1	0	1	0			(N)
65353	0	1	0	0	1	0	0	1	**LF-ON**		(T,L)
65369	0	1	0	1	1	0	0	1			(N)
65350	0	1	0	0	0	1	1	0	**HF-ON**		(T,L)
65366	0	1	0	1	0	1	1	0			(N)
65381	0	1	1	0	0	1	0	1	**SR**	(+ PG)	(T,L,N)
65386	0	1	1	0	1	0	1	0		(+ SG)	(T,L,N)
65385	0	1	1	0	1	0	0	1		(+ LF)	(T,L,N)
65382	0	1	1	0	0	1	1	0		(+ HF)	(T,L,N)
65389	0	1	1	0	1	1	0	1	**SR rep**	(+ PGLF/LFPG)	(T,L,N)
65383	0	1	1	0	0	1	1	1		(+ PGHF/HFPG)	(T,L,N)
65387	0	1	1	0	1	0	1	1		(+ SGLF/LFSG)	(T,L,N)
65390	0	1	1	0	1	1	1	0		(+ SGHF/HFSG)	(T,L,N)
65509	1	1	1	0	0	1	0	1	**RW-ON**	(+ CONF-PG)	(T,L,N)
65514	1	1	1	0	1	0	1	0		(+ CONF-SG)	(T,L,N)
65513	1	1	1	0	1	0	0	1		(+ CONF-LF)	(T,L,N)
65510	1	1	1	0	0	1	1	0		(+ CONF-HF)	(T,L,N)
65504	1	1	1	0	0	0	0	0	**RW rep**	(+ LED c)	(T,L)
65440	1	0	1	0	0	0	0	0			(N)
65376	0	1	1	0	0	0	0	0	**GO-OFF/RW-OFF**		(T,L,N)
65312	0	0	1	0	0	0	0	0	**STOP**		(T,L,N)
65391	0	1	1	0	1	1	1	1	**ERROR**	(+ table switch)	(T,L)
65359	0	1	0	0	1	1	1	1			(T,L)
	reward pump	LED c	table switch	trial start	LED bl	LED tr	LED tl	LED br			

Table 2.4 – Overview of digital trial events. Translation table for the 8-bit binary code created by LabView for the activation (bit status 1) or deactivation (bit status 0) of the LEDs of the cue system, the table switch, the reward pump or the by LabView internally set trial start. The most left column shows the NSP translation of the 8-bit binary code into a decimal code that was stored with time stamp in the nev-file. The decimal code is actually calculated from a 16-bit version of the 8-bit binary code where the last 8 bits are all set to 1 (e.g., 65296 for TS-ON is actually calculated from the binary code 1111111100010000). The second column from the right shows the event interpretation of the codes in a trial context. Due to an update of the LabView control program the digital event codes differ between the monkeys. Which code occurs in which monkey is marked in the most right column. Some signals originate from a mistake in the sampling of the digital events by the LabView program (SR rep and RW rep). They can occur sporadically in the nev-file and can be ignored in the context of a trial.

2.4.1 Digital event translation

performance interpretation	occurrence (1) or non-occurrence (0) of ... STOP	RW-ON	SR	GO-ON	CUE-OFF	CUE-ON	WS-ON	TS-ON	pc
incomplete trial	0	0	0	0	0	0	0	0	**0**
error occurred before SR	1	0	0	1	1	1	1	1	**159**
error (SR) occurred before WS-ON	1	0	1	0	0	0	0	1	**161**
error (SR) occurred before CUE-ON	1	0	1	0	0	0	1	1	**163**
error (SR) occurred before CUE-OFF	1	0	1	0	0	1	1	1	**167**
error (SR) occurred before GO-ON	1	0	1	0	1	1	1	1	**175**
grip error	1	0	1	1	1	1	1	1	**191**
correct completed trial	1	1	1	1	1	1	1	1	**255**
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
	(8-bit) binary codes								

Table 2.5 – Overview of trial performance codes. From the translated trial events listed in Table 2.4 8 consistent digital events were chosen (see header) to validate the completeness of a trial reflecting the performance of the monkey. In an incomplete trial either a TS-ON or a STOP event was not detectable. All other additionally occurring events were neglected in the context of trial performance. The occurrence (1) or non-occurrence (0) of the 8 chosen digital events were used to create the 8-bit binary code versions (middle part of table) of the later on transformed decimal performance code (pc, most right column). The interpretation of in context of trial performance is stated in the leftmost column. The pc is universal between the monkeys.

Table 2.4 lists the $8-bit$ combinations that were sent by LabView to activate or deactivate the physical devices of the experimental apparatus. These 8-bit combinations were also sent to the NSP where they were first transformed to a $16-bit$ representation with the last $8 bits$ set to one and secondly converted to a corresponding code of decimal numbers (cf. Table 2.4). The latter was saved along with the time stamps in the nev-file. In a first preprocessing step, these decimal codes needed to be translated to a human-readable format and put into context of an expected trial event sequence. The validation against the latter was used in addition to identify incomplete, correct and error trials. For this 8 consistent trial events were picked and their occurrence (1) or non-occurrence (0) was used to create an artificial $8-bit$ binary code for each trial to describe its completeness (cf. Table 2.5). This generated binary code again was once more converted to a decimal number defining the performance code (pc) for each validated trial, with pc 255 representing a correct completed trial (cf. Table 2.5). With this method, it was possible to classify error types and distinguish an error trial where the monkey used the wrong grip from error trials where the monkey just moved too early (cf. pc 191 for grip error vs. the remaining performance codes for error trials in Table 2.5).

To facilitate and standardize the correct data access of trials among all collaboration partners, I implemented the digital event translation, and performance interpretation, which are respectively summarized in Table 2.4 and Table 2.5, as automatically conducted functions within the experiment specific data loading routine (described in Section 2.5.2.2).

2.4.2 Behavioural event extraction

Some behavioural events, such as the object touch (OT) or the object displacement onset (DO) by the monkey were monitored during the experiment to monitor the used grip type and pull movement of the monkey (cf. Section 2.3.2), but their online-detected timing was only approximate. For this reason these measurements were not immediately saved with the recorded data, but were calculated

more accurately later on offline from the analogue signal traces of the four FSR sensors reflecting the monkey's grip (GFpr1, GFpr2, GFside1, and GFside2, cf. Table 2.3) and the HE sensor recording the object displacement (Displ, cf. Table 2.3). For this, a custom-made Matlab toolbox was implemented by a technician of the CoMCo group (Dr. F. Jaillet) to detect the following eight specific behavioural events for each trial:

OT the precise timing of object touch from one FSR sensor trace

OR the precise timing of object release from one FSR sensor trace

DO the precise timing of displacement onset of the object from the HE sensor trace

OBB the precise timing when the object returned back to its baseline from the HE sensor trace

FSRplat-ON the precise timing of the onset of the plateau phase of one FSR sensor trace

FSRplat-OFF the precise timing of the offset of the plateau phase of one FSR sensor trace

HEplat-ON the precise timing of the onset of the plateau phase of the HE sensor trace

HEplat-OFF the precise timing of the offset of the plateau phase of the HE sensor trace

As indicated, the toolbox uses only one trace of the four FSR sensors. The selection is first justified by the requested grip type of each trial which reduces the FSR sensor traces to either GFpr1 and GFpr2 for PG, or GFside1 and GFside2 for SG (cf. Table 2.3). In addition, typically one of the two grip force traces is better qualified to measure OT, OR, FSRplat-on, and FSRplat-off, which leads to the final trace selection for the behavioural event detection.

The toolbox can perform an automatic detection of the listed events, where their timing is first approximated by crossing a manually set threshold for the corresponding traces and then fine-tuned by back-comparison of the traces with baseline level from the point of threshold crossing. Nonetheless, the automatic detection was prone to errors, so that the trials were visually inspected one by one and the timing of the automatically detected events were manually corrected if they did not match the event times as visually identified. The detected behavioural events mark different movement components performed by the monkey in each trial and can become relevant during further data analysis. In this context, OT marks the grip onset of the monkey, while DO provides the actual start point where the monkey pulls the object. The timing of OR and OBB provides the information whether the monkey released the object immediately after the requested holding period, or if the monkey tended to push the object back into its original place. FSRplat-ON and HEplat-ON as well as FSRplat-OFF and HEplat-OFF can be used to check for how long the monkey steadily hold the object in a constant position.

Besides the detection of the listed behavioural events, a second Matlab script was used to inspect the load force trace (LoadForce, cf. Table 2.3) in each trial to control if the actual object load corresponded to the programmed object load. This procedure ensured that the electro-magnet controlling the object load was properly activated throughout the recording session.

2.4.3 Offline spike sorting

The spike waveforms, which were extracted and pre-sorted online during a recording via Central Suite to gain an overview of the quality of the spiking activity, were (re)sorted offline using the Plexon Offline-Sorter (version 3.3.3). To keep the variability in this half-manual spike sorting at a minimum, all sorting processes were performed by the same person (Dr. A. Riehle). The spike sorting started

with loading the nev-file of a recording into the Plexon Offline Sorter. Here, the file is internally duplicated, and the spike sorting performed on this duplication, keeping the original file intact. The sorting started by joining all different waveforms extracted online from each channel separately back again into one pool (unit 0 in the nev-file, "unsorted waveforms" in Plexon sorter) ignoring the results of the preliminary online waveform sorting (unit 0-16 in the nev-file). For the invalidation of cross-channel artefacts (e.g. chewing artefacts), all waveforms that occurred simultaneously on a defined number of channels (70%) were extracted from the pool and stored as noise (unit 255 in nev-file, "invalidated waveforms" in Plexon sorter) of the corresponding channels. Furthermore, a waveform-rejection was performed. Thereby all waveforms of abnormally large amplitude and/or atypical shape on a channel were manually extracted from the pooled waveforms and also stored as noise (unit 255 in the nev-file, "invalidated waveforms" in Plexon sorter). These signals classified as noise were not included in the following cluster analysis.

For the spike sorting cluster analysis, the remaining waveforms were processed individually per each channel. The actual spike sorting was done by using different algorithms splitting the remaining waveforms in clusters in an 2 or 3-dimensional principal component (PC) space. The dimensionality of the PC space was chosen according to the best separation. The main cluster algorithms used were K-Means-Scan or Valley Seeking (chosen according to the best separation). For both algorithms a fixed outlier threshold (a parameter to be determined in the Plexon sorter) was used, which was set to a value of 1.8 for K-Means-Scan and 2 for Valley Seeking to get comparable sorting results. The spikes of the sorted clusters were then controlled using the inter-spike-interval (ISI) distributions and the auto and cross-correlation plots. Units were ordered manually from best to worst (assigning increasing unit ids 1-16 in nev-file and Plexon sorter) by considering the amplitude of the waveform (the higher the better), the outcomes of the ISI (no or low number of spikes with an ISI smaller than 2 ms), the correlation histograms, and identifiable cluster shapes. Clusters with unacceptable outcomes (completely or partly overlapping waveforms), including those with only a few spikes, were classified as unsorted (channel 0 in the nev-file, "unsorted, waveforms" in Plexon). In the end, the duplicated nev-file contains the time stamps and waveforms of now sorted single or multi-units (unit 1-16) and remaining noise (unit 0 or unit 255). Note that unit 255 contains invalidated signals, whereas unit 0 contains signals that may enter a further cluster analysis for spike sorting. This offline spike sorted nev-file was saved under the file name of the original nev-file with an added tag (e.g. -01). The nev-file with the sorted units can be loaded again into the Plexon Offline Sorter (even without a license) to access all the chosen sorting parameters.

The half-manual approach of the offline spike sorting by a single researcher is a rather time consuming process. For this reason, not all recorded datasets of the R2G experiment are resorted offline yet. Today (Wednesday 30$^{\text{th}}$ November, 2016), 27 datasets of the first and 23 datasets of the second recording period of monkey T (cf. Table A.6 and Table A.7), 72 datasets of monkey L (cf. Table A.9 and Table A.8), as well as 11 datasets of the first and 27 of the second recording period of monkey N were offline spike sorted (cf. Table A.10 and Table A.11). The averaged numbers of identified single and multi units (SUA and MUA) of these offline spike sorted datasets varied between the implanted arrays. Figure 2.9 shows the (summed) number of SUAs that were identified on each electrode of the array across all (so far) offline spike sorted datasets. Especially the results for the first recording period of monkey T show that the quality of the spiking activity was very poor.

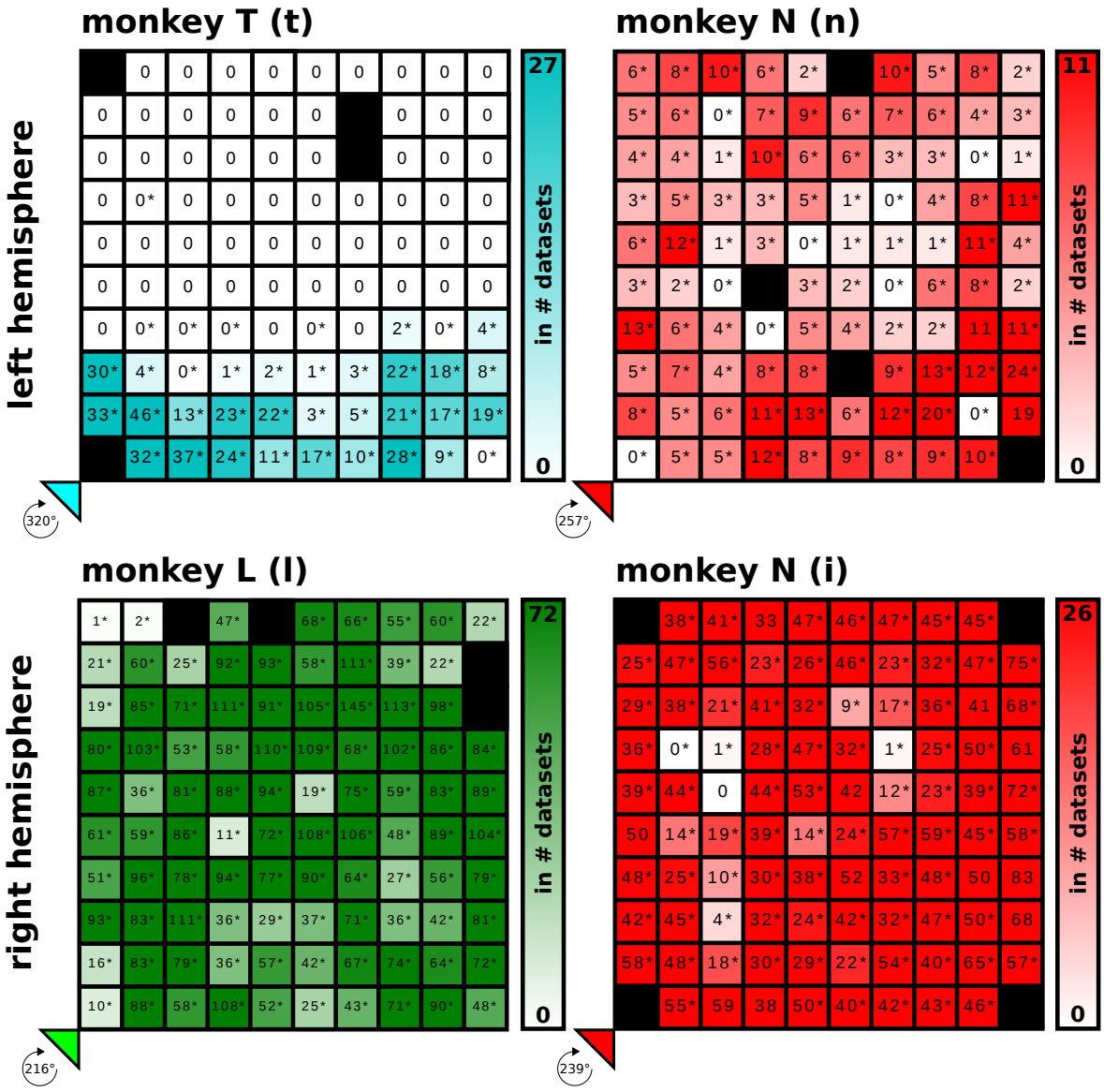

Figure 2.9 – Overview of identified single and multi unit activities. The figure displays the number of identified SUAs and MUAs across each array (numbers and * displayed for each electrode, respectively) summed over all (so far) offline spike sorted datasets (colour bar, effective: Wednesday 30$^{\text{th}}$ November, 2016) from the first recording period of monkey T (t, in cyan), monkey L (l, in green), and monkey N (n, in red), as well as from the second recording period of monkey N (i, also in red). The number of identified SUAs in relation to the number of spike sorted datasets (intensity of colour of each electrode) reflects the quality of the recorded spike data for each electrode over time. Recordings from light coloured or white shaded electrodes show little spiking activity in most datasets, which could mean that the corresponding electrodes either were of not sufficient quality from the beginning, or were damaged during the implant procedure, or are located in the wrong cortical layer due to a slanted implantation of the array. The latter was presumed to be the reason for the little spiking activity recorded from the array of monkey T(t). For the electrode configurations of each array see Figure 2.8. For the implant location of each array compare with Figure 2.2 and Figure 2.3.

2.4.4 Quality assessment

The occurrence of noise in electrophysiological recordings is to a certain degree unavoidable and therefore needs to be carefully examined. It depends to a large extent on the quality of the headstage used to record the neurophysiological data. In the reach-to-grasp experiment, two different types of headstages were used for the two monkeys, the Samtec-CerePort headstage (monkey L) and the Patient Cable (monkey N). The former is much more sensitive to noise than the latter. Nonetheless, in both cases, the type of noise, its cause and appearance in the data is quite variable. Depending on the direct influence of the different types of noise on subsequent analysis methods, one needs to balance the corresponding data rejection between being very permissive and very conservative. For this reason, it is wise not remove or delete data of bad quality, but instead mark them with the judgement of a corresponding quality assessment procedure. This approach not only allows the user to finally decide which data to reject for an analysis, but also provides the opportunity to provide different quality assessments of, e.g., the same electrode, trial and unit at the same time. This is especially helpful if one considers that certain types of noise can differently contaminate signals in different frequency bands. For the R2G experiment, the quality of the recorded signals was separately tested for the sorted spike data and different frequency bands of the LFP data. The used corresponding procedures are described in detail below.

2.4.4.1 Correction of data alignment

Due to an error in the Blackrock DAQ system, the ns6-file always starts 82 samples later than a parallel recorded ns5, ns2 or nev-file. Furthermore, the online filtered LFP signals in the ns2-file are delayed by 3.6 ms with respect to the time stamps in the nev-file because of the online filter procedure. Note also that the time stamps of the spike times provided in the nev-file correspond to the online threshold crossing time of the raw signal (10 time stamps after the waveform start). All expected time shifts between the different data files were hard-coded in implemented data loading routines (Section 2.5.2), which then automatically corrected the data alignment.

2.4.4.2 Data gaps

During a preliminary spike triggered average (STA) analysis of the data of monkey L, it was realized that at some time point in the recording, data packages of an ns5-file were lost causing unreported gaps in the continuous neuronal analogue signals of all channels. After further investigation it was clear that these data package losses happened in nearly all ns5-files of monkey L, and in possibly two ns5-files of monkey T (see Table 2.6).

The degree of damage this system failure had on the usability of ns5-file datasets becomes most clear in the description of its discovery. In an STA analysis, recorded spike times are used to extract a certain time window around each spike of a simultaneously recorded continuous signal, and the resulting pool of signal snippets is averaged in the end. With this, the STA analysis is commonly used to study the temporal relationship between a spike train and a simultaneously recorded continuous signal (Ito, 2014). The method offers a suitable initial analysis for the spike trains of the offline sorted units and the simultaneously recorded analogue signals from the electrodes of the Utah-Array in the R2G experiment. If the STA analysis is conducted on a spike train and analogue signal recorded from the same electrode, the STA should display a spike component at t_0. When this preliminary analysis was conducted by a student of the SN group (J. Sprenger) on the R2G datasets of monkey T and L, the student discovered an unexpected shifted spike component in the results which led to the discovery of the package loss.

monkey T

	recording				data gap			
filename	**task**	**cnd**	**seq**	**dur**	**det**	**#**	**@**	**dur**
t090310-001	?			344s	pos			
t100310-001	2-inst	cnd11	gt:bl	425s	no	0	-	-
t100310-002	2-inst	cnd11	gt:bl3	461s	pos			
t120310-001	2-inst	cnd11	gt:bl3	953s	no	0	-	-
t120310-002	2-inst	cnd11	gt:bl3	797s	no	0	-	-
t120310-003	?			35s	-	-	-	-
t290410-003	2-inst	cnd1	ra	315s	no	0	-	-
t030510-001	2-inst	cnd1	ra	441s	no	0	-	-
t030510-002	-	-	-	1s	-	-	-	-
t040510-001	2-inst	cnd1	ra	471s	no	0	-	-
t050510-001	2-inst	cnd1	ra	446s	no	0	-	-
t100610-001*	2-inst	cnd1	ra	740s	no	0	-	-
t100610-002*	2-inst	cnd1	ra	651s	no	0	-	-

monkey L

	recording				data gap			
filename	**task**	**cnd**	**seq**	**dur**	**det**	**#**	**@**	**dur**
l101208-001	2-inst	cnd1	ra	313s	no	0	-	-
l101208-002	2-inst	cnd1	ra	668s	yes	1	580s	10ms
l101208-003	1-inst	cnd14	ft:bl1	320s	yes	1	250s	10ms
l101210-001	2-inst	cnd1	ra	709s	no	0	-	-
l101210-002	2-inst	cnd2	ra	732s	yes	2	90s	50ms
							500s	30ms
l101213-001	2-inst	cnd1	ra	702s	yes	1	510s	50ms
l101213-002	2-inst	cnd1	ra	813s	yes	1	270s	80ms
l101213-003	2-inst	cnd2	ra	118s	no	0	-	-
l101213-004	sleep	-	-	114s	no	0	-	-
l101213-005	sleep	-	-	144s	yes	1	50s	8ms

Table 2.6 – Overview of data gaps caused by a package loss in the ns5-files. The table lists for each ns5-dataset of monkey T and L (see filename) if a package loss occurred (det: yes/no), and if yes, the number (#), time point (@), and duration (dur) of the resulting data gaps in the neuronal analogue signals of the recording (effective: Wednesday 30$^{\text{th}}$ November, 2016). Due to low quality of the recorded spiking activity, the execution of this quality assessment procedure proved to be difficult in monkey T and the corresponding results need to be treated carefully. It seems that, compared to monkey L, a possible package loss (pos) only occurred in two datasets, but further analysis is necessary to confirm or decline the suspicion. The recording duration of t120310-003 and t030510-002 were too short to be analysed, but are also not useful for any study. As additional information for all datasets of both monkeys, the corresponding task, condition (cnd), trial type sequence (seq) and duration (dur) of the recording is also listed. The * marks datasets for which only selected channels were saved in an ns5-file. gt:blX and ft:blX mean that the grip types or force types are alternated in blocks of X trials, respectively. For further abbreviations and codes of the task and cnd entries see Section 2.1. Note that in monkey T, for the datasets t090310-001 and t120310-003 the task paradigm could not have been determined (marked with ?), because the corresponding metadata could not have been retraced so far.

Further investigation of the matter revealed that the old version of the Cerebus DAQ system used in monkey T and L could not report this package loss, because the analogue data packages were not provided with a time stamp of the continuously running common clock. Even worse, due to this fact the analogue signal after the lost packages was just concatenated to the end of signal before. In contrast, the spike data recorded in the nev-file were provided with timestamps from the continuously running clock of the system, which resulted in a correct time line and only missing spikes due to the data loss. For this reason, the analogue data of an ns5-file were misaligned with respect to the spike data after the first package loss leading to the inconsistent results in the STA analysis. Unfortunately, the package loss happened at arbitrary time points and for an arbitrary number data packages in nearly all ns5 datasets of monkey T and L, so that the gap detection could not be automatized, but had to be an interactive time consuming process which will be described in the following.

In this quality assessment, each dataset of monkey T and L in which the raw analogue signals recorded from the electrodes of the Utah-Array were stored in an ns5-file has to be examined for the potential occurrence of a package loss. If a data package was lost, corresponding data were lost on all electrodes. This fact facilitated the detection procedure to be performed on one selected electrode for which high quality spiking activity data existed. To detect if a package was lost the following steps were conducted. First, an STA analysis was performed, usually on the last $100\,sec$ of the recorded data from the selected electrode. The actual duration of the time window that was investigated depended obviously on the recording length of the corresponding data. Second, if a shift in the spike related component of the STA was observed, packages were lost at least once and the exact time point(s) of the system failure needed to be investigated in more detail. For this, the STA analysis was repeated on different time windows of the recorded data to narrow down the actual time point of the package loss. Once a smaller time window of the incidence was determined, the "precise" time point of the package loss and the duration of the resulting gap in the data needed to be measured manually. For this, the data snippet of the analogue signal of the selected electrode around the potential package loss was visualized in parallel to the data snippet of a corresponding spike train from the same electrode. The time point of the package loss was then set to the last spike that visually was aligned to the corresponding deflection in the analogue signal. The duration of the resulting gap in the data was estimated by determining as precisely as possible the amount of time the spike times after the last aligned spike needed to be shifted, in order to match the corresponding deflections of the analogue signal again. The results of this quality assessment step (effective: Wednesday 30$^{\text{th}}$ November, 2016) were saved for each monkey in a txt-file and are summarized in Table 2.6 and source 17 in Section 2.5.1.

In the updated version of the Cerebus DAQ system used in monkey N, the cause of the frequent system failure was corrected by the manufacturer and the data packages of the analogue signals were provided with their corresponding time stamps, so that a potential package loss would have been reported by the system.

2.4.4.3 LFP data

The LFP data were examined for noise in three broad frequency bands excluding the 50Hz European line noise (low frequency components, LFC: $3\,Hz - 10\,Hz$, intermediate frequency components, IFC: $12\,Hz - 40\,Hz$, high frequency components, HFC: $60\,Hz - 250\,Hz$) in each session individually. The goal of the quality assessment was, first, to detect channels with a noisy signal throughout the session and, second, to detect noisy trials in the remaining "clean" channels. To do so, I implemented a module in Python which conducts the following quality assessment of electrodes and trials.

To start the quality assessment of the LFP data, the analogue signal of each electrode was first

z-scored and filtered in the three frequency bands (LFC, IFC, and HFC) using a Butterworth filter (of order 2, 3, and 4, respectively). The quality assessment of electrodes and trials was carried out separately for each frequency band. The detection of noisy electrodes was performed first, executing the following three steps:

step 1 The variance of the filtered analogue signal of each electrode was calculated over the complete session.

step 2 Out of the 96 resulting variance values, outliers were identified as those values outside a user-defined range. The range was defined as follows: (i) values between a lower (e.g., 25^{th}) and an upper (e.g., 75^{th}) percentile (L and U), (ii) the range of acceptable values was defined by

$$[L - w \cdot (U - L), U + w \cdot (U - L)], \tag{2.1}$$

where w is a user-defined whisker coefficient (e.g., $w = 3$).

step 3 The analogue signals classified as outliers in step 2 were visually controlled by comparing them to the analogue signal of an electrode with a typical variance value. If the results were either too conservative or too permissive, the detection procedure was repeated by manually adapting the chosen parameters (L, U, and w), correspondingly.

The electrode IDs of the final outliers were marked as *noisy* for the tested frequency band and saved in an hdf5-file along with the chosen parameters of the quality assessment analysis of the corresponding recording.

The detection of noisy trials was subsequently performed on the remaining non-noisy electrodes. The quality assessment of trials was carried out in a procedure analogue to that for detecting noisy electrodes, with the following difference: the variance of the filtered analogue signal was calculated for each trial on each non-noisy electrode separately. At the end, the trial IDs of the identified outliers were pooled and marked as *noisy* for the tested frequency band on all electrodes. Note that the analysis for detecting noisy trials indirectly depends on the prior detection and exclusion of noisy electrodes. The underlying reason is that the results of the trial quality assessment are affected by single electrodes, because a trial is marked as noisy on all electrodes as soon as it is classified as noisy on one electrode. It is therefore important to generally exclude noisy electrodes before conduction the quality assessment analysis of trials. Due to the dependency, the marked trial IDs and corresponding analysis parameters were saved in the same hdf5-file of the previously executed quality assessment of electrodes for the same tested frequency band.

2.4.4.4 Spike data

To test and judge the quality of the spike data, the results of the offline spike sorting were controlled first for the signal-to-noise ratio (SNR) from the waveforms of the identified single units and second for the occurrence of hyper-synchronous event artefacts.

1. For each spike sorting of the R2G experiment, the SNR for each identified unit was calculated in a Matlab program written by the experimental partners, complying a method introduced by Hatsopoulos et al. (2004). In this method, the SNR is defined as the amplitude (A, trough-to-peak) of the mean waveform ($< w >$) divided by twice the standard deviation of the waveform

noise (SD_{noise}) of the defined unit (u):

$$SNR_u = A_{<w>}/SD_{\text{noise}} \cdot 2, \tag{2.2}$$

where SD_{noise} is computed by averaging the standard deviations (SDs) obtained from each sample point across the original waveforms (SD of the waveform noise adapted from Nordhausen et al. 1996 and Suner et al. 2005). For the identified single units in the datasets of the R2G experiment, the determined SNRs typically ranged between 1.5 and 12. Corresponding to Suner et al. (2005), the quality of the spike sorting of an identified unit is highly adequate if the SNR is above 4, fair if the SNR ranges between 2 and 4, and poor if the SNR ranges between 1 and 2. Units with an SNR below 1 are not considered as signals. For a conservative analysis of the spike datasets, only single units with a $SNR \geq 2.5$ were used in published research of the R2G experiment, such as in Torre et al. (2016). The parameters and results of the SNR analysis of the controlled spike sorting were saved in an mat-file of the corresponding recording. The quality assessment of the single units was not yet conducted for all the current available offline spike sorted datasets (cf. table A.6 – A.11).

2. Since correlation analysis of spike data is very sensitive to cross-electrode artefacts which would produce unwanted false positive results, the sorted spike data were controlled on their original time resolution ($\delta = 1/30 ms$) for potential occurrence of hyper-synchronous event artefacts. For this, a population histogram was computed, i.e. the sum of the spikes across all sorted single units in the dataset in bins of $\delta = 1/30 ms$ (sampling resolution of the data) was calculated, and checked for hyper-synchronous spike events (bin with≥ 2 spikes). Surprisingly, such hyper-synchronous spikes, which are likely to be attributed to cross-channel noise, survived the spike sorting including the cross-channel artefact removal by the Plexon Spike sorter. The occurrence of these spike artefacts was discovered during a preliminary analysis of a previous study of the R2G experiment (Torre et al., 2016). Further investigation of the matter showed that the number of single units participating in such events can range from 2 to over 30, and a statistical analysis showed that the frequency of their occurrence largely exceeded the expected value considering the observed population firing rate. Moreover, a $\delta-$binned time histogram of the population spiking activity triggered around the occurrence times of the hyper-synchronous events also revealed increased spiking activity in the preceding or following bin of the event. For a conservative analysis of the spike datasets, it is therefore recommendable to treat the spikes participating in a hyper-synchronous event as well as the spikes occurring within a short time interval around this event ($\pm\delta$) as artefacts of unknown origin, and to remove them subsequently before performing any analysis of the spike data.

In Torre et al. (2016) both quality assessments of the spike data were combined, considering only spikes with a $SNR \geq 2.5$ and additionally removing all hyper-synchronous events with ≥ 2 synchronous spikes.

2.5 Metadata and data accessibility

Although a data descriptor does provide a comprehensive documentation of the background story, workflow, and peculiarities of an experiment, it is additionally necessary to guarantee the accessibility and usability of the corresponding data and metadata to successfully share them across different levels (cf. Section 1.1). For this reason, data journals, such as Scientific Data, request, in addition to the descriptor and the provision of data and metadata on an appropriate public repository: (i) a machine-

readable metadata file, (ii) a statement indicating whether and how code, that was implemented to record, load or pre-process the data and metadata, can be accessed, and used. In the following subsections, I will explain why these two requests by data journals are not only reasonable, but highly beneficial across all data sharing levels. In this context, I will introduce how I implemented these requests for the R2G experiment.

2.5.1 Machine-readable metadata - organization matters

In the R2G experiment, metadata were originally captured in the laboratory notebook(s), the manual of the used DAQ system, and the factory information sheets of the array manufacturer. In addition, some information about individual recordings were transferred into a huge Excel spreadsheet, but this listing was neither complete nor the entries standardized. Other metadata were captured in the original data files of each recording, but not necessarily accessible by the available loading routines at that time. For this reason, the first step towards creating a comprehensive metadata collection was to reorganize and complete the metadata sources.

As a first result, metadata sources of the R2G experiment were distributed over various files and formats, but standardized in each file into machine-readable key-value pairs. Figure 2.10 provides a schematic overview of all metadata source files including the original, only human-readable metadata sources (shaded in black). In the corresponding list of source files (s) below, I summarize the actual content of each source and how it was generated.

s1) Laboratory notebook(s): The laboratory notebooks were (in case of the new monkey E: are) generated by hand containing metadata about the setup (e.g., what version of DAQ system was used), the monkeys (e.g., the body weight before each surgery), and the recordings (e.g., the conducted task) from the beginning until the end of the R2G experiment. The information is handwritten by the researchers conducting the recordings (Dr. T. Brochier, Dr. Y. Hao, Dr. G. Prabhu, M. Zaeppfel, M. Duret, and S. Wirtssohn) and remains with the PI of the experiment (Dr. T. Brochier). The laboratory notebook(s) contain crucial information to understand what was recorded when. For this reason, it is essential to transfer the metadata into a machine-readable format (cf. source 5, 6, and 7, as well as source 9).

s2) Cerebus manual: The current Cerebus user manual, available as pdf-file via the manufacturer, provides for all available DAQ system versions all hard and software specifications and the generic internal structure of the binary data files. Some metadata needed to be manually extracted from this documentation of the Cerebus DAQ system, because, especially for the older version of the DAQ system, several (default) metadata about the structure and selected parameter of the recorded data, such as the data unit of the recorded analogue signals, were not saved in a machine-readable format along with the data. Nonetheless, these metadata are essential for the basic understanding of the data and are even necessary to correctly load them. For this reason, the corresponding metadata were manually transferred into a machine-readable format (cf. source 7 and 9).

s3) Utah-Array configuration sheet: The Utah-Array configuration sheet is provided by the manufacturer Blackrock Microsystems with each sold array either as xls-spreadsheet or pdf-file. It contains metadata describing the wiring and corresponding configuration of electrodes of the corresponding array (cf. Figure 2.8), including the factory impedance of the electrodes. Unfortunately, the Utah-Array configuration sheet is not perfectly standardized and not always

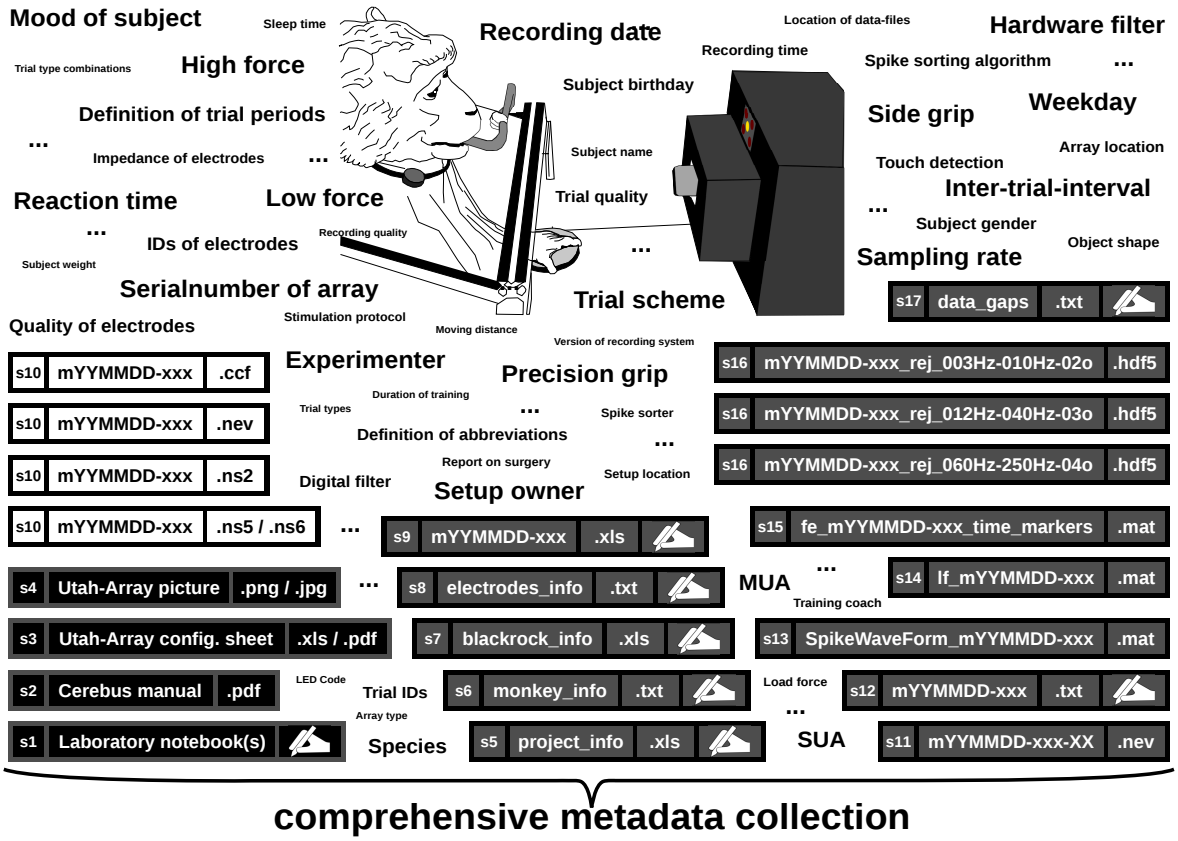

Figure 2.10 – Metadata sources of the reach-to-gasp experiment. The R2G experiment contains a large amount of various metadata as indicated by the word cloud around the R2G setup sketch. All these metadata were captured in one of the labelled source files (sX) listed below the word cloud. Besides the generalized filename of each source file, also the format of the file is stated and a writing hand indicates if the source was manually created. What metadata the source files contain is explained in the main text (cf. labels), but the white and grey shaded source files are all machine-readable, while the black shaded source files are not. Metadata contained in these four only human-readable files were transferred by hand into a machine-readable form in one of the source files grouped centrally. The other source files are further grouped into files containing data (white shaded group), and files generated in preprocessing steps (four groups on the left, from top to bottom: gap correction, quality assessment of LFP data, behavioural event and force detection, offline spike sorting with corresponding quality assessment of the spike data). Of all source files only the `project_info` file is valid for all recording periods across all monkeys. When which source file is created or updated is described in the main text and schematically displayed in Figure 3.1. To facilitate the access to the metadata of the R2G experiment, it is best to compile all these source files into a comprehensive machine-readable metadata collection, as indicated at the bottom of the figure. In case of the R2G experiment, the odML framework was used to compile this collection (cf. Section 1.2 and Figure 3.6).

provided in a machine-readable format. For this reason, the corresponding information was transferred by hand into a machine-readable, standardized txt-file (cf. source 8).

s4) Utah-Array picture: The location of the Utah-Array after implantation was captured as photography during the surgery (cf. Figure 2.2 and Figure 2.3) and stored along with other files containing information on the corresponding Utah-Array. The anatomical landmarks as they are marked in the figures are determined by eye, as well as the degree of rotation of the array for the implantation. The latter was then entered as metadata value into a machine-readable spreadsheet (cf. source 6).

s5) Project information spreadsheet: In the beginning, I created in collaboration with the experimenters a project information spreadsheet which contained metadata that were valid across all datasets of the R2G experiment. This included general abbreviations, codes or descriptions of constant hardware components (e.g., cf. task and trial type abbreviations in Section 2.1). These metadata were originally captured in the laboratory notebooks and then manually transferred into a machine-readable key-value representation in an xls-spreadsheet. To simplify the data and metadata management workflow, these constant metadata were directly implemented as parameters in the data loading routines, and the generator script used to compile a comprehensive metadata collection.

s6) Monkey information spreadsheet: For each recording period of each monkey, I created in collaboration with the experimenters a monkey information spreadsheet. This spreadsheet contained profile metadata of each monkey (e.g., species, identifier, gender, birthday, cf. Section 2.2), key information on the task training (e.g., start and end date, cf. Section 2.2.1) and on the array implant surgery (e.g. date, hemisphere, array rotation, cf. Section 2.2.2). As for source 5, these metadata were originally captured in the laboratory notebooks and then manually transferred into a machine-readable key-value representation in an xls-spreadsheet.

s7) Blackrock information spreadsheet: For each recording period of each monkey, I created in collaboration with the experimenters a Blackrock information spreadsheet that contained metadata describing the type of the used Utah-Array and its connector (e.g., serial number, geometry of electrode grid, number of active electrodes, connector type, etc.) and general information about the hard and software components of the DAQ system which remained unchanged during the recording period. The corresponding information was collected mainly from the Cerebus user manual (source 2) and manually transferred into a machine-readable key-value representation in an xls-spreadsheet.

s8) Electrodes information spreadsheet: As previously mentioned, metadata contained in the Utah-Array configuration sheet provided by the manufacturer Blackrock Microsystems were not easily extractable (see source 3). For this reason, the experimental group extracted the DAQ system IDs and corresponding uniform caIDs (cf. Figure 2.8) of each electrode from this source and manually created a corresponding, but unlabelled tab separated list stored as txt-file. Based on this, I extended and modified by hand the corresponding txt-file to a labelled tab separated list providing the caID (column 1), the bank and pin ID of the Front-End-Amplifier of the Cerebus DAQ (column 2 and 3), the DAQ system ID (column 4), the factory impedance in$k\Omega$ (column 5), and the grid ID (column 6).

s9) Recording information spreadsheet: As I mentioned in the first paragraph of this subsection, information on individual recordings (e.g., conducted task, task condition, sequence order of trial

types, cf. Section 2.1) was originally captured as handwritten notes in the laboratory notebooks. It was then partially transferred per recording period to a huge Excel spreadsheet, listing in each row the metadata for one recording, partially as key-value pairs or as written notes. When I started to extend and standardize the entries of this Excel overview spreadsheet to a complete set of recording-specific metadata for all available datasets for each recording period, the size of the document soon became confusing and difficult to maintain. For this reason, I reorganized in collaboration with the experimenters the corresponding metadata as machine-readable key-value representations into one spreadsheet per recording. This recording-specific spreadsheet included all metadata which could have been modified in a single recording, but were not saved automatically, such as the headstage, the conducted task paradigm, the task settings (condition, trial type sequence, used object loads, cf. Section 2.1), the file settings to save the neural and behavioural signals (e.g., nsX type and corresponding filter settings, cf. Section 2.3.3), as well as key information on the performed offline spike sorting (if it was already conducted, cf. Section 2.4.3).

s10) Original data files: As described in the data descriptor (cf. Section 2.3.3), the original data files were automatically saved for each recording and contained a set of recording specific metadata, such as the date of the recording day, the start time of the recording, the sampling rates of the stored signals, as well as the the time stamps for the coded digital events reflecting the activation or deactivation of devices and the analogue signals recorded from the object sensors reflecting the monkey's trial behaviour. The Python loading routine for these files which were available at the time I started working on the project did not provide access to all these metadata. For example, the old loading routine did not provide access to events and type of error trials. In the new loading routine(s), that I implemented in collaboration with two other members of the SN-group (Dr. M. Denker and J. Sprenger) using the Neo data framework, all metadata available in the original data files were used to annotate the corresponding Neo data objects.

s11) Data file of offline spike sorting: As described in the data descriptor (cf. Section 2.4.3), the data file that contained the resorted unit IDs resulting from the offline spike sorting unfortunately neither contained the metadata information about the classification of unit as SUA or MUA nor provided information on how the offline spike sorting was conducted. The latter was, as already mentioned, manually captured in the recording specific information spreadsheet (cf. source 9). The classification of units as SUA and MUA was instead stored in an unlabelled tab separated txt-file (cf. source 12).

s12) Offline spike sorting information sheet: As described in the data descriptor (cf. Section 2.4.3), to state if a resorted unit is classified as SUA or MUA, the responsible researcher (Dr. A. Riehle) manually created for each conducted offline spike sorting an unlabelled tab separated txt-file. This file listed for each electrode (DAQ system ID, column 1), the number of units that were classified as SUAs (column 2), and the unit ID of the unit containing remaining MUA (column 3). The actual unit IDs for the SUAs were deducible from their count, because they consecutively increased starting with ID 1. The generation of this file was, however, error prone. For this reason, I implemented in collaboration with Dr. M. Denker a loading routine for this source file that included several sanity checks for the manually entered values, such as controlling if each entered electrode ID only occurred once, and if the listed MUA ID was actually one value higher than the listed number of SUAs.

s13) Data file of offline spike sorting quality assessment: As described in the data descriptor (cf. Section 2.4.4.4), after an offline spike sorting was performed, the waveforms of the resulting resorted units were analysed by the experimenters via a custom made Matlab script to identify the quality of the sorting. The results and selected parameters of this quality assessment procedure were stored in a mat-file. In detail, this file contained for each identified single unit on each electrode the number of recorded spikes, the corresponding waveforms, the averaged peak-to-peak amplitude of the waveforms, the averaged spike duration, and three differently calculated SNR values, of which SNR3 was calculated as described in the data descriptor.

s14) Data file of load force extraction: As described in the data descriptor (cf. Section 2.4.2), some time after the recording was conducted, for each trial, the determined load force of the target object (LF, HF or none if the grasping behaviour was not executed) was extracted from the corresponding FSR sensor. This procedure was executed by the experimenters via a custom made Matlab script that stored the corresponding results of this preprocessing step in a mat-file. In this file, the results were listed for each trial as tuple, where the first entry corresponds to the trial ID (starting from 1) and the second entry coded the detected load force, using 0 for LF, 1 for HF and -1 for none.

s15) Data file of behavioural event extraction: As described in the data descriptor (cf. Section 2.4.2), some time after the recording was conducted, for each trial, the eight events, OT, FSRplat-ON, FSRplat-OFF, and OR, as well as DO, HEplat-ON, HEplat-OFF, and OBB, which characterize the grasping behaviour of the monkey, were extracted from the corresponding sensors of the target object (FSR grip sensors and HE sensor, respectively). This procedure was executed by the experimenters via a custom made Matlab module that stored the corresponding results in a mat-file. In detail, this file contained for each trial type a list of corresponding trial IDs, a second list of trial IDs of all trials that were manually double-checked, a statement which FSR grip sensor signal was used for the detection of OT, FSRplat-ON, FSRplat-OFF, and OR (1 or 2), and, for each trial, a list of the extracted times for all eight behavioural events (in the same order as stated above). In this file, the trial types are coded, using 1 for SGHF, 2 for SGLF, 3 for HFSG, 4 for HFPG, 5 for LFSG, 6 for LFPG, 7 for PGHF, and 8 for PGLF.

s16) Data files of LFP quality assessment: As described in the data descriptor (cf. Section 2.4.4.3), recorded LFP data were generally examined for noise in three broad frequency bands marking either the complete signal or only trials from an electrode as unusable. The parameters and results for this quality assessment procedure for each frequency band were saved as a Python dictionary in a hdf5-file. A Python dictionary is structured (similar to an odML-file) in hierarchically organized key-value pairs. The metadata for the selected filter settings (`filtername` to abbreviate the specific settings (e.g., LFC, IFC, or HFC), the selected `highcut` and `lowcut` frequency (in Hz), as well as the selected filter `order`) are saved under a common key (`fsets`) at the top level of the file. The parameters (used `quartiles`, used `whis`, and resulting `threshold`) and results of the detection of noisy electrodes (electrode IDs organized in `bad_electrodes` and `good_electrodes`) and the subsequent detection of noisy trials (trial IDs organized individually for each electrode in `bad_trials4electrodes`, as well as pooled across the array in `bad_trials` and `good_trials`) are saved on the same under the keys `elrej` and `trrej`, respectively.

s17) Data files of data gap detection: As described in the data descriptor (cf. Section 2.4.4.2), a tab separated txt-file lists for all ns5-files of one monkey if a data package was indeed lost, and

if yes, when it occurred and for how long the resulting gap in the data lasted. The list structure of this txt-file is comparable to the structure of Table 2.6 for the individual monkey.

In summary, there is (i) one machine-readable source file that contains metadata valid across all datasets of the R2G experiment, (ii) three source files that contain separated information about the monkey before each recording period, the general settings used for the Cerebus DAQ system for each recording period, and the electrode configurations of the Utah-Array used in each recording period, (iii) five source files per recorded dataset that contain information on the used adjustable parameter settings of the recording system, the conducted task and corresponding settings, as well as potential observations of the experimenter obtained during the recording, and (iv) nine metadata source files containing parameters and results of the preprocessing steps conducted to technically validate each dataset (cf. Section 2.4). With this, for each datasets of the R2G experiment, metadata are distributed over 15 files and 8 formats. Even files of the same format cannot necessarily be loaded in the same way due to differences in the structure of the content. In addition, in some files codes were used to describe existing/generated metadata creating new metadata that are necessary to understand the content of the corresponding files.

To facilitate the access to all these metadata, it was decided that the information contained in the machine-readable source files should be compiled into a new comprehensive, more standardized collection using the odML metadata framework (cf. Section 1.2). With this, the metadata file sources were reduced to one odML-file per recorded dataset which contained all information necessary to understand and work with the corresponding data. Such a comprehensive odML collection of condensely stored and standardized metadata facilitated not only the metadata exchange between the collaboration partners of the R2G experiment and the provision of machine-readable metadata for a potential data publication (cf. Section 2.6), but also provided the opportunity to more easily connect and enrich loaded datasets with corresponding metadata (cf. Section 2.5.2). The latter optimized the workflow and increased the reproducibility of analysis projects performed on the data of the R2G experiment (cf. Chapter 2). In order to access the metadata stored in the an odML-file, the corresponding library of the API `python-odML` can be used. A tutorial on how to utilize this library can be found in the online documentation shipped with the library[3], and a more detailed description of how I managed the metadata of the R2G experiment by example of the odML framework can be found in Chapter 3 (which is based on Zehl et al., 2016). This chapter also includes tutorial like code examples on how to generate and use odML-files in Python and Matlab. In short, the library supports opening an odML-file, reading its contents into an library-specific representation, and uses API functions for filtering the corresponding hierarchical structure, and extracting key-value pairs of interest. Thus, the API `python-odML` provides a convenient, standardized way to access the metadata records.

2.5.2 Data access

The datasets of each recording of the R2G experiment consists of two parts. First, primary data are provided as the original data files obtained from the Central Suite software stored in the data format specified by the manufacturer (in particular, nev, ns5 and ns6 format) of the recording hardware, Blackrock Microsystems. Second, metadata are provided as one file in the odml-format (Grewe et al., 2011). As such, all files are provided in generic formats that are well documented in the Cerebus manual (cf. source 2 in Section 2.5.1) in the case of the primary data, and in Grewe et al. (2011) as well as on the G-Node projects website[4] in the case of metadata (cf. Section 1.2).

[3] https://github.com/G-Node/python-odml

[4] http://www.g-node.org/projects/odml

Nevertheless, mere knowledge of the generic internal structure of the files is insufficient to make practical use of the provided data. The reason is twofold: first, primary data and metadata need to be merged in a meaningful manner that is specific to the R2G experiment. Second, if the task of developing the code to load the data is left to individual researchers, both the complexity of the data and the file formats are likely to provide incoherent and error-prone implementations. In the following, I provide a working solution based on the Python programming language to read the datasets, and demonstrate how this approach can be used to utilize the data in other programming languages. In short, I will first use openly available libraries to separately load primary data (see `BlackrockIO` in Section 2.5.2.1) and metadata (cf. Section 2.5.1 and Chapter 3) into a defined structure based on the Neo data and the odML metadata framework (cf. Section 1.3 and Section 1.2, respectively). In a second step, I implement an experiment specific loading routine (see `ReachGraspIO` in Section 2.5.2.2), that attaches further relevant metadata extracted from the odML-file as annotations to the Neo data objects hosting the primary data. If required, the annotated Neo data structure that hosts a complete representation of the data and corresponding metadata may be saved to disk in a serialized manner in a standardized container format (e.g., mat or hdf5-file), such that it can be accessed from other programming languages, such as Matlab.

Complete link collections to the `python-neo` and `python-odML` libraries can be found at the NeuralEnsemble[5] and G-Node[6] website, respectively. Importantly, both projects are hosted and version-controlled via Github[7][8][9]. Updated versions of the `ReachGraspIO` and all example code can also be found on Github[10].

2.5.2.1 Task 1: providing a generic BlackrockIO for Neo

In case of the R2G experiment, the data object model provided by the open source Neo library (Garcia et al., 2014) was chosen as the primary representation of the datasets (cf. Section 1.3). However, the corresponding I/O module for the file format used by Blackrock Microsystems (class `BlackrockIO` in file `neo.io.blackrockio.py`) was unfortunately not fully functional. For this reason, as my first task, I re-implemented and extended the available `BlackrockIO` in collaboration with other members of the SN-group (Dr. M. Denker and J. Sprenger). The corresponding code was reintegrated into the `python-neo` library and is now publicly available again.

In the new I/O module, I used two previous versions of the `BlackrockIO` to combine the optimized, efficient, and more Neo compliant code of iteration 3 by S. Garcia with the important extended functionality of iteration 2, a first attempt from Dr. M. Denker and myself to update the module. In addition, I added several new features to the module, to support data files with specifications 2.1, 2.2 and 2.3 that originate from different versions of the Cerebus DAQ system. In this context, I also restructured the code of the `BlackrockIO` to make it easier to add upcoming file specifications of the DAQ system by grouping read operations common to multiple file specifications, termed variants. For example, the function `BlackrockIO.__read_nsx_header_variant_a` is used to load the header of a provided nsX-file with the specification 2.1, while the function `BlackrockIO.__read_nsx_header_variant_b` is used to load the header of a provided nsX-file with the specification 2.2 or 2.3. To provide as much support as possible for the user not to have to worry about the version differences of the data files

[5] http://neuralensemble.org/neo/

[6] http://www.g-node.org/projects/odml

[7] https://github.com/about

[8] https://github.com/NeuralEnsemble/python-neo

[9] https://github.com/G-Node/python-odml

[10] https://github.com/INM-6/reachgrasp-dataset

of the Cerebus DAQ system, the file specification is extracted automatically at the beginning of the implemented loading process, determining which variant function or setting will be used and providing a common output with a data object structure that is independent of the version of the Cerebus DAQ system.

In the following, I summarize the output for a dataset of the R2G experiment obtained when calling the I/O module with default parameters. The `BlackrockIO` returns a Neo Block object as a top level grouping object. In the hierarchy directly below the Block is one single Segment object spanning the complete continuous recording, and one `ChannelIndex` object for each of the 96 electrodes of the Utah-Array. If all data files are available, the analogue signals recorded from each of the 96 electrodes and the 6 object sensors are each saved in one `AnalogSignal` object. All of these `AnalogSignal` objects are linked to the `Segment` object. The `AnalogSignal` objects containing data recorded from an electrode are in addition linked to the corresponding `ChannelIndex`. Likewise, the spike times (and optionally, the spike waveforms) of each identified unit of the original nev-file are saved to a `SpikeTrain` object. If instead the units of the nev-file resulting from the offline spike sorting should be used, the corresponding filename has to be explicitly stated for the key-word argument `nev_override` when first calling the `BlackrockIO`. As for the `AnalogSignals`, the `SpikeTrain` objects are linked to the Segment and corresponding `ChannelIndex` objects. Finally, all digital events are saved into a single Neo `Event` object that lists their time of occurrences and the corresponding event codes. Additional information which can be immediately extracted from the original data files is provided via the Neo object attributes and annotations.

Note that although this generic I/O can be used to access any data records originating from a Cerebus DAQ system, no experiment-specific interpretation of the file contents is given. For example, digital events are not interpreted in context of a reach-to-grasp trial, but only labelled with the codes generated by the DAQ system (cf. Section 2.3.2, Section 2.3.3, and Table 2.4).

2.5.2.2 Task 2: annotate Neo data objects with more metadata

The task of combining the primary data obtained from the Neo `BlackrockIO` module and the metadata obtained from the generic loader of the API `python-odML` is performed by a custom-written Python module named `RGIO`. For this, the `ReachGraspIO` class implemented in the `RGIO` module was derived from `BlackrockIO` class of the corresponding Neo module, to initially load the primary data with the publicly available generic routine. In the following, I summarize the modifications of the output for a dataset of the R2G experiment obtained when calling the `ReachGraspIO` module with default parameters.

In short, the `ReachGraspIO` performs the following steps: (i) read the primary data using the parent class (`BlackrockIO`), (ii) read the metadata using the `python-odML` library, (iii) interpret event data based on the digital events (e.g., detect trial start or reward, cf. Section 2.4.1 and Table 2.4), (iv) integrate the extracted behavioural events stored in the odML-file into the `Event` object, (v) add relevant metadata of the odML-file as annotations to the corresponding Neo data objects.

The resulting output of the `ReachGraspIO` is in its hierarchical Neo object representation of the primary data equivalent to the generic output returned by the `BlackrockIO`, but enriched with additional metadata information in order to allow the user to interpret the data and extend them with results of the experiment specific technical validation procedures.

2.6 What is going to be published in *Scientific Data*

In summary, the R2G experiment started over 9 years ago and currently contains over 1500 datasets collected during five recording periods performed by three monkeys. A fourth monkey is currently in training which will provide new datasets from two further recording periods in the coming year. Re-collecting information on the available datasets of the R2G experiment to generate the data descriptor of the R2G experiment in its current state, including the implementation of a comprehensive data and metadata framework (cf. previous sections of Chapter 2), was accordingly a time consuming process. As of today, three research paper have been published by the collaboration partners that are based on datasets from the R2G experiment (Riehle et al., 2013; Milekovic et al., 2015; Torre et al., 2016). Three additional research paper are going to be published in the near future (e.g., Denker et al., prep). Nonetheless, the potential for further publications is high if one considers not only the sheer amount of data of the experiment, but also all the different tasks and conditions which were executed and of which only a few were examined in the current publications. Considering the invested time and effort invested in conducting the R2G experiment and in setting up a proper data sharing workflow between the collaboration partners, it is completely understandable that collaboration partners do not want to leave the fruits of these potential publications to be collected by unknown third parties. However, the collaboration partners also share the general opinion that publicly releasing data for the use by others is beneficial for scientific progress. For this reason, it was decided to publish one dataset from monkey L (l101210-001) and one dataset from second recording period of monkey N (i140703-001) in a data journal, such as Scientific Data. The selection of only these two datasets was founded on the following three arguments (arg):

arg 1 Publish datasets with unprocessed content. The argument was to publish datasets that contained the raw signals recorded from the Utah-Array, to provide third parties the opportunity to extract potential spike data by other methods than the one used via the Cerebus DAQ system in the R2G experiment. This reduced the possible datasets of monkey T and L to a minimum, because raw data were only stored for 12 and 10 recordings, respectively.

arg 2 Avoid publishing corrupted datasets. The argument developed over the discovery of the system failure causing the package loss in the raw data recordings of monkey T and L. These corrupted datasets are difficult to handle and interpret, and further testing is necessary to guarantee that the realignment procedure of the data signal is accurate enough to not affect subsequent analysis results. For this reason, the number potential datasets for publication of monkey T and L were reduced to X and 4, respectively.

arg 3 Publish datasets with acceptable and comparable recording quality. This argument is the weakest, but valid if one considers that, for a data publication, one has to provide datasets that can be meaningfully used by third parties. Datasets containing signals of significantly bad quality undermine this request. As demonstrated in Figure 2.9, the datasets of monkey T in general showed very little spiking activity and only on roughly one third of the electrodes of the Utah-Array, which points to an unevenly insertion of the array into the cortex during the implant surgery. For this reason, it was decided to not publish any datasets from monkey T. The same reason applied to the datasets from the first recording period of monkey N. Furthermore, two datasets (l101208-001 and l101213-003) of the remaining 4 for a potential publication of monkey L were excluded, because the monkey performed far too few trials in them.

In summary, the arguments excluded datasets of monkey T's and monkey N's first recording period completely for publication and reduced the number of potential datasets for monkey L to two. In one

of these two, monkey L actually did not perform the standard reach-to-grasp task, but was sleeping (l101213-004). For monkey N the sleeping behaviour was never recorded, which led to the decision to publish only one dataset from monkey L (l101210-001) and one dataset from the second recording period of monkey N (i140703-001). Although the reduction of over 1500 datasets to two datasets seems preposterous, the decision was well thought over and the usefulness of the two datasets for promoting research progress in the community is still high, which I will summarize in the following three paragraphs.

The two datasets are useful to study interactions in the cortical network. The high quality of the two datasets provide a large number of simultaneously recorded single neurons (93 and 156, for L and N, respectively) along with the continuous neuronal raw signals (sampled at 30kHz and broadly band-pass filtered to 0.3Hz - 7.5kHz). In addition, a down-sampled and filtered version of the raw signals is provided for monkey N and can be computed from the data of monkey L, which allows researchers to study the local field potential (LFP) (Mitzdorf, 1985; Logothetis and Wandell, 2004; Einevoll et al., 2013). For these reasons, these high-dimensional parallel datasets, although only two are going to be published, still provide the opportunity for neuroscientists and computational neuroscientists to study interactions in the cortical network during different epochs of a well described behaviour.

The two datasets are useful to study network coordination in high-dimensional data and develop corresponding analysis methods. To date, not many tools enable to study network coordination in high-dimensional data, since non-experimentalists, such as statisticians or theoretical/computational neuroscientists do not often have access to such data. Nevertheless, methods for correlation analysis of high-dimensional data that do not run into a combinatorial explosion and have acceptable computing times are highly needed. This also implies the strong need for methods and tools that perform dimensionality reduction to reduce the complexity of the data and the computational load (e.g., Cowley et al., 2013). The development of such analysis methods requires knowing the typical features of experimental data, such as e.g. non-stationarities in time and across trials, as well as deviations from typical theoretical assumptions (e.g. Poisson), in order to make the methods applicable to experimental data. If these features are ignored and not considered in the method, there is a considerable danger of generating false positive outcomes and potentially wrong interpretations of the results (Grün, 2009; Louis et al., 2010). The datasets that will be published allow non-experimentalists to get insights in features of cortical data from awake behaving animals.

The two datasets are useful to test and validate spike sorting methods. Our data also provide the possibility to test and validate spike sorting methods (see Denker et al., 2014a). Besides the raw data, we publish the corresponding spike data resulting from an offline spike sorting using a Plexon-Offline-Sorter. Other sorting methods can be applied to the same (raw) data, and differences in the results can be compared and analysed.

To facilitate the summarized potential usage of the datasets by third parties, (i) the primary data will be provided as original data files obtained from the Central Suite software stored in the data format specified by the manufacturer (in particular, nev, ns5 and ns6 format) of the neural recording platform, Blackrock Microsystems, (ii) an offline sorted version of the neural spike data will be provided in a second nev-file (cf. Section 2.4.3), (iii) metadata containing all information about each dataset including our parameters and results of the described preprocessing steps will be

provided as one file per dataset in the odML format (Grewe et al., 2011; Zehl et al., 2016), and (iv) a mat-file will be provided containing the continuous neural raw data together with the offline sorted spike data, both annotated with the corresponding metadata. Furthermore, all code required to access the data as described in Section 2.5.2 is stored along with these datasets. The provided code includes, in particular: (i) a snapshot of the Python Neo package, (ii) a snapshot of the Python odML package, (iii) and the custom-written `ReachGraspIO` extending the generic Neo `BlackrockIO` module. With the publication of a data descriptor (cf. Chapter 2) specified for these two datasets in the journal Scientific Data, all files of the datasets as listed above and the corresponding code to access them will be publicly available via the data portal of the German Neuroinformatics Node (G-Node) of the International Neuroinformatics Coordination Facility (INCF), called GNData[11].

As of today (Wednesday 30$^{\text{th}}$ November, 2016), these two datasets will be the first published datasets of massively parallel recordings from monkeys performing a complex instructed delayed reach-to-grasp task. For more information on the selected datasets see Section "Overview of R2G datasets that are going to be published" in the Appendix.

[11] http://g-node.github.io/g-node-portal/

Chapter 3

Metadata management

The following chapter is based in main parts on the "Handling Metadata in a Neurophysiology Laboratory" manuscript which was published in Frontiers of Neuroinformatics (Zehl et al., 2016).

As one can imagine from the complexity of the exemplary data descriptor I presented in Chapter 2, properly managing metadata of such an experiment is not a trivial task. The relevance of some metadata does not immediately become apparent and is sometimes underestimated (cf. development of knowledge space in Section 1.2). For example, the immediate relevance of each minute detail of the experimental setup are of little interest for the interpretation of a single recording, and only when one attempts to rebuild the experiment do these metadata become valuable. Apart from that, any piece of metadata can become highly relevant for screening and selecting datasets according to specific criteria or to comprehend the occurrence and origin of artefacts discovered in the data at a later stage. Different experiments produce different sets of metadata and it is therefore not possible to provide a detailed list of metadata that needs to be collected from any given type of experiment. Nevertheless, the general principle should be to keep as much information about an experiment as possible from the beginning on, even if information seems to be trivial or irrelevant at the time. This is the only approach to prevent the loss metadata, caused by a lack of knowledge at the time that these metadata my become important later on (cf. development of knowledge space in Section 1.2). Based on the experience I gathered as member of a laboratory involved in various collaboration projects, I compiled a list of guiding questions to help other scientists defining their optimal collection of metadata for their use scenario, such as reproducing the experiment and/or the data analysis, or for data sharing (cf. data sharing levels in Section 1.2):

- What metadata would be required to replicate the experiment elsewhere?
- Is the experiment sufficiently explained, such that someone else could continue the work?
- If the recordings exhibit spurious data, is the signal flow completely reconstructible to find the cause?
- Are metadata provided which may explain variability (e.g., between subjects or recordings) in the recorded data?
- Are metadata provided which enable access to subsets of data to address specific scientific questions?

- Is the recorded data described in sufficient detail, such that an external collaborator could understand them?

For the R2G experiment presented in Chapter 2, I used these guiding questions to evaluate whether the content of the various metadata sources was sufficient, and enriched these where necessary. The process of planning the relevant metadata content and to (re)organize the metadata into a comprehensive collection can be time-consuming, especially if the process happened after the experiment was performed, as it was the case for the R2G experiment. In the following, I will first outline why the time for creating a comprehensive metadata collection is, nevertheless, well spent (Section 3.1), secondly, summarize how to compile of such a collection and what needs to be considered in the process (Section 3.2), and at last, demonstrate in a tutorial-like manner how to make use of such a metadata collection (Section 3.3).

3.1 Why make the effort to create a comprehensive metadata collection?

The time and effort I invested in (re)organizing metadata of the R2G experiment to compile a comprehensive collection using an established metadata framework was immense, and the process to cover all datasets of the R2G experiment in this collection is not yet completed. If a corresponding metadata management is planned in advance of the experiment, instead of being first implemented afterwards as was the case for the R2G experiment, the investment of time and manpower will be considerably less intense. The storage of metadata using a corresponding framework can be automatized from the beginning, reducing the number of metadata sources to a minimum, and hidden knowledge in form of handwritten notes or implicit knowledge of the experimenter can be avoided and do not have to be recollected in a cumbersome interrogation process. Nonetheless, the required planning process and implementation of a proper metadata management will remain a time-consuming procedure.

Based on the example experiment (cf. Chapter 2), I present now four cases that demonstrate in which metadata become important and why the time for creating a comprehensive metadata collection is nevertheless well spent. For the implementation of each case, I contrast the situation before and after having organized metadata in a comprehensive collection:

Situation 1: Metadata are organized in different files and formats which are stored in a metadata source directory.

Situation 2: Metadata sources of Situation 1 are compiled into a comprehensive collection, stored in one file per recording using a standard human and machine-readable file format and being accessible and editable with standard software tools.

As described in Section 2.5.1, in the course of reorganizing the metadata management of the R2G experiment, I personally experienced both situations while working with the data. For the initial work with the data, I started with complementing the contents of the different metadata source files given the various formats. This work uncovered that handling the different metadata sources while working with the data was too difficult and error prone. This motivated the change of metadata organization and recompiling the sources into a comprehensive metadata collection with one metadata file per dataset, transferring Situation 1 into Situation 2. The following four example cases describe these difficulties I had in Situation 1 in more detail, which then led to the decision to implement a comprehensive metadata collection.

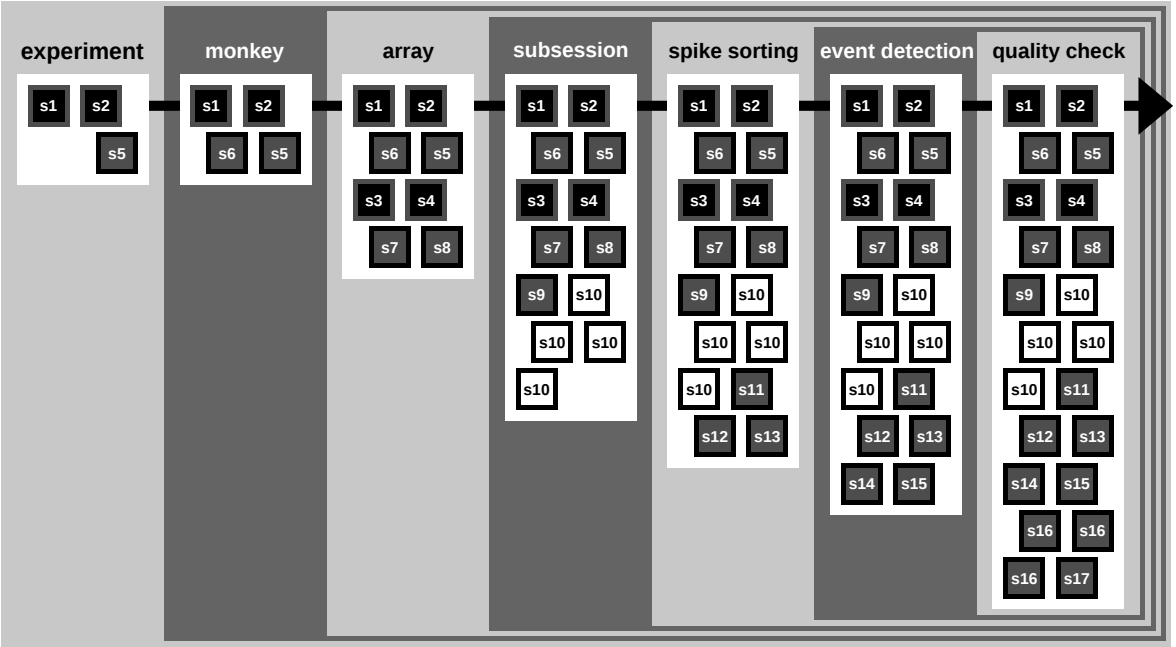

Figure 3.1 – Emergence of metadata sources over time in the R2G experiment (modified figure 2 of Zehl et al., 2016). Small grey and labelled boxes represent the different source files which contain metadata (cf. Section 2.5.1). In the beginning of the experiment, the metadata collection first contains only a source with information that stays constant for the complete project. With each monkey and recording period, the metadata collection is enlarged by sources containing monkey-specific information. With each array implantation, sources with information about that array are included in the metadata collection. During or directly after each recording, the metadata collection is extended by sources containing information specific to the corresponding recording. Several technical validations are then performed on the recorded data (here summarized as: spike sorting, event detection, and quality check) and the corresponding selected parameters and results are also integrated as metadata into the collection.

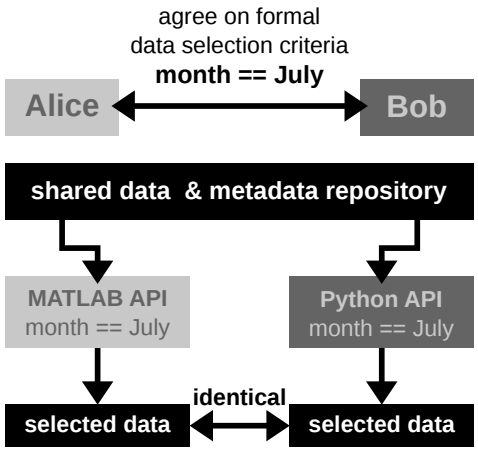

Figure 3.2 – Example of a formalized metadata base data selection. (modified figure 4 of Zehl et al., 2016) Schematic workflow of selecting data across laboratories based on available APIs for a comprehensive metadata collection. In this optimized workflow, researcher A (Alice) and researcher B (Bob) are both able to identically select data from the common repository by screening the comprehensive metadata collection for the selection criteria (`month == July`) via the available APIs for MATLAB and Python. Such a formalized query in data selection supports a well-defined collaborative workflow across laboratories.

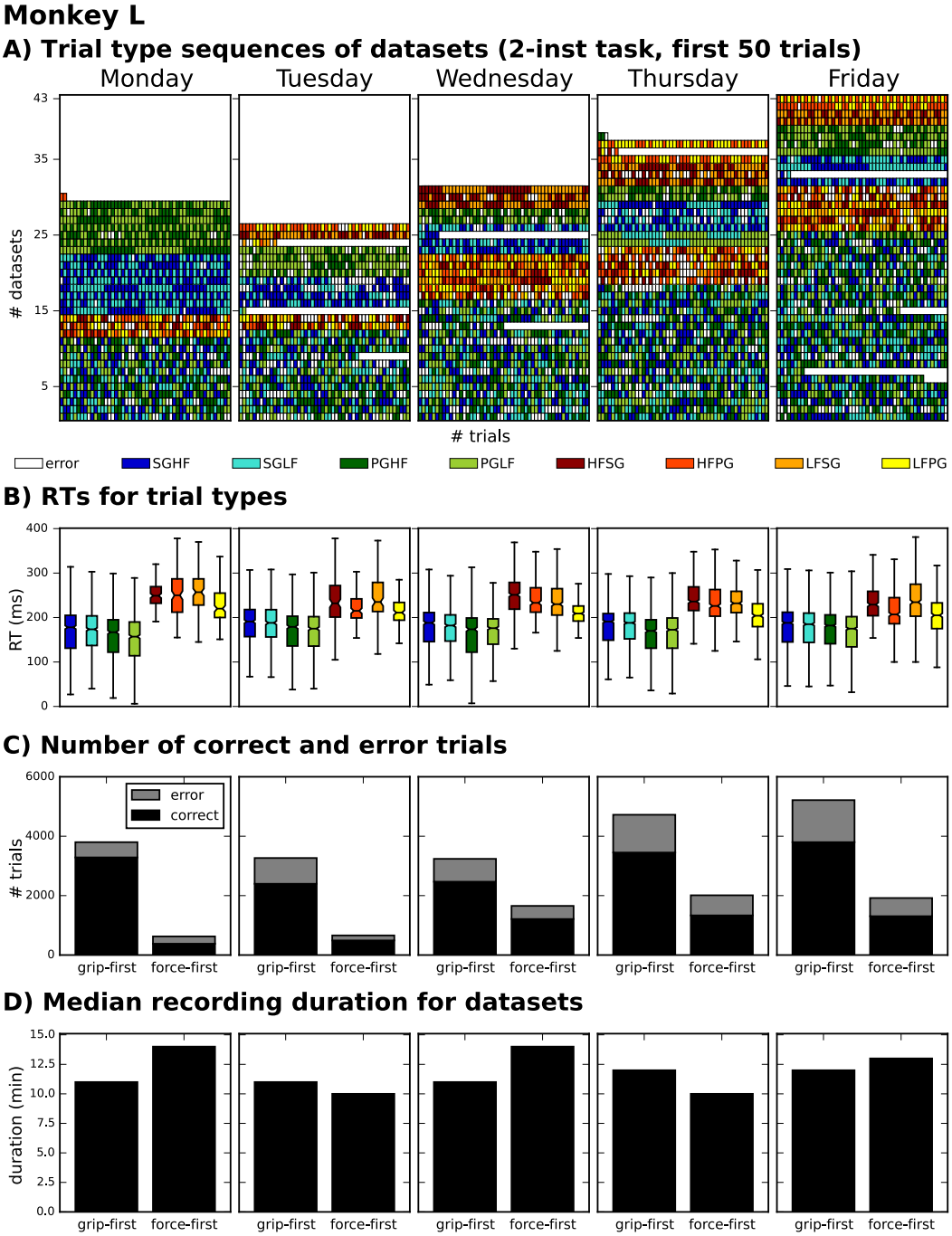

Figure 3.3 – **Example of a metadata analysis.** (modified figure 3 of Zehl et al., 2016) The figure provides an example of an overview of metadata summarizing the performance of monkey L on each weekday in the R2G experiment. **A)** Generic display of the order of trial sequences of the first 50 trials (small bars along the x-axis) for each dataset for the standard task (rows along the y-axis) recorded on the corresponding weekday. Each small bar corresponds to a trial and its colour to the requested trial type of the trial (see colour legend below sub-plot A; for trial type acronyms see Section 2.1.1). **B)** Summary of the median reaction times (RTs) for all trials of the same trial type (see colour legend) of all datasets for the standard task recorded on the corresponding weekday. **C)** Bar-plots for each weekday that display the total number of trials of all datasets for the standard task differentiated between correct and error trials (coloured in black and grey) and between grip-first and force-first task conditions (cf. Section 2.1.1). **D)** Bar-plots for each weekday that show the median recording duration of the grip-first and force-first task conditions of the corresponding datasets for the standard task.

Case 1) Metadata maintenance and exchange. Metadata originate at different time points during an experiment. For example, in the R2G experiment, information on the monkey's gender or which array was implanted is known early on, information which task is conducted in which recording is first known on the recording day. Furthermore, the technical validations, such as necessary preprocessing steps to gain more information from the data (e.g., offline spike sorting) or quality assessments of the data, are usually performed some time after the recording took place. If not explicitly handled differently from the beginning of an experiment, the natural emergence of metadata over time automatically results in Situation 1 (cf. Figure 3.1), where metadata are distributed over several files and formats with different internally structured content (cf. original metadata sources of the R2G experiment in 2.10 and Section 2.5.1). Maintaining and understanding the variety of the usually unstandardised metadata sources, especially over longer time periods, is difficult even for a single researcher, not to mention for potentially external members in a research collaboration. In Situation 2, a comprehensive collection bundles metadata together, and simplifies access by combining them into one single, standard file format. With this, a comprehensive metadata collection has the advantage not only to simplify the maintenance of metadata for a single researcher or research laboratory (cf. data sharing level A and B in Section 1.3), but also to facilitate the metadata exchange with external collaboration partners or third parties (cf. data sharing level C and D in Section 1.3). Hence, an easily maintainable and exchangeable metadata collection, combining, e.g., parameters and results of important preprocessing steps, increases the reproducibility of conducted research across different data sharing levels.

Case 2) Human-readable metadata access. Keeping an overview of what was conducted in an experiment is an important aspect, not only for an experimentalist during and after a running project, but also for a theoretician who is working on the data. Gaining such an overview is often achieved by manually inspecting the metadata in a human-readable form. Compared to a human-readable comprehensive metadata collection in Situation 2, a manual inspection of different metadata sources in Situation 1 has the following disadvantages: (i) it is difficult to keep track of which metadata source contains which piece of information; (ii) not every metadata source is readable with a simple, easy accessible software tool (e.g., binary files, see source 10 in Section 2.5.1); (iii) not every software tool offers the option to search for a specific metadata content. For these reasons, a human-readable, standardized and searchable organization of the metadata in a comprehensive metadata collection is highly beneficial, because it guarantees a complete and easy access to all metadata related to an experiment even over a long period of time. This simplifies the long-term comprehensibility an experiment not only for a single researcher, but also for all further group members and collaboration partners (cf. data sharing level A, B, and C in Section 1.3).

Case 3) Machine-readable metadata access. Gaining an overview of what was conducted in an experiment can also require a more analytical method across the metadata of an entire experiment. In such cases it is crucial that metadata are machine-readable and can be extracted from their collection in an automatic screening procedure. A nice and simple example for this case in the R2G experiment would be the generation of an overview figure of selected metadata from datasets for the standard task, separated by the weekdays of the corresponding recording days, to investigate if the weekend break negatively influenced the performance of a monkey (see Figure 3.3). The corresponding metadata screening process is negatively influenced in Situation 1 if (i) metadata are distributed over several files and formats, especially if (ii) some are stored

in the original data files, and if (iii) secondary metadata first need to be computed from primary metadata (e.g., in the reach-to-grasp tasks of the example experiment the reaction times (RTs) need to be calculated from the the events GO and SR, cf.). All three aspects slow down the screening procedure and complicate the code to extract the corresponding metadata. For the comprehensive metadata collection in Situation 2, the workflow to automatically extract metadata is improved, because (i) all metadata (primary and secondary) are stored in one file per dataset, and (ii) the format of the metadata files is standardized and accessible via a publicly available, generic loading routine. The reduction of metadata files to one per recording without the overload of potentially attached primary data reduces the run time of the screening procedure. In addition, both improvements have the advantage that the implemented code for the metadata screening procedure is less complicated. which makes the implementation and its output easier to understand and reproduce for all group members, collaboration partners, and third parties (cf. data sharing level A, B, C, and D in Section 1.3).

Case 4) Data analysis based on metadata. Besides the usefulness of performing standalone metadata analysis across datasets of an experiment, metadata are also frequently used to select datasets according to certain criteria to, e.g., avoid corrupted datasets, or constrain data to answer specific scientific questions. Such criteria are usually represented by metadata related to one recording and emerge directly from settings of the recording or from subsequently conducted technical validation procedures. A common metadata based data selection is aggravated if (i) collaboration partners base their work on different workflow strategies and software technologies (e.g., different programming languages), and if, in addition, (ii) a geographical separation represents a communication barrier that requires extra care in conveying relevant information to the partner in a precise way. For this, metadata should be termed as concrete formal specifications with a clear link to the corresponding data. Compared to Situation 1, metadata in Situation 2 are stored in a standardized format for each dataset that reflects the structure of defined key-value pairs. For example, the recording date of each dataset is stored in key-value pairs of the corresponding metadata file providing the `year`, `month`, `day`, and `weekday` of the recording. By using such a standardized metadata specification, a query for datasets, which, e.g., were recorded in July (`month == July`), can be handled via available APIs of the used metadata framework. Therefore, the query is guaranteed to produce the same result for all scientists, even if they use different programming languages (see Figure 3.2). The formalization of metadata in key-value pairs in a standardized, comprehensive metadata collection will result in a more coherent and less error-prone synchronization between the work of different laboratories. In addition, such a metadata formalization is more compatible with data frameworks that allow researchers to annotate data objects with important metadata (cf. usage of the Neo data framework in the R2G experiment Section 2.5.2).

3.2 Management strategies for a comprehensive metadata collection

The four cases illustrate the importance and usefulness of a comprehensive metadata collection in a standardized format. As introduced in Section 1.2, for the R2G experiment, I chose to accordingly manage metadata using the odML metadata framework, which was proposed by Grewe et al. (2011) and is maintained as an open-source project by the G-Node. My decision for this metadata framework was guided by the requirements I define in Section 1.2. The definition of these requirements is

```
import odml

# generate an odML Value
odml_value_1a = odml.Value(data='Macaca mulatta',
                           dtype=odml.DType.string)

# generate an odML Property containing the generated odML Value
odml_property_1a = odml.Property(name='Species',
                                 value=odml_value_1a,
                                 definition='Binomial species name (genus,'
                                            ' species within genus)')

# generate another odML Value
odml_value_1b = odml.Value(data=4.9,
                           dtype=odml.DType.float,
                           unit='kg')

# generate an odML Property containing the generated odML Value
odml_property_1b = odml.Property(name='Weight',
                                 value=odml_value_1b,
                                 definition='Body weight of the subject.')


# generate an odML Section
odml_section_1 = odml.Section(name='Subject',
                              type='subject',
                              definition='Information on the investigated'
                                         ' experimental subject (animal or'
                                         ' person)')

# group the two generated odML Properties into generated odML Section
odml_section_1.append(odml_property_1a)
odml_section_1.append(odml_property_1b)

# generate another odML Section containing another Property-Value pair
odml_value_2 = odml.Value(data='2011-09-30',
                          dtype=odml.DType.date)

odml_property_2 = odml.Property(name='Date',
                                value=odml_value_2,
                                definition='Date of the surgery')

odml_section_2 = odml.Section(name='ArrayImplant',
                              type='subject/preparation',
                              definition='Information on the array implant'
                                         ' performed on subject')

odml_section_2.append(odml_property_2)

# group second odML Section as subsection into the first odML Section
odml_section_1.append(odml_section_2)

# generate an odML Document
odml_document = odml.Document(author='Bob')

# group first odML Section with all children into generated odML Document
odml_document.append(odml_section_1)

# save the odML Document as odML file
save_to = 'subject_information.odml'
odml.tools.xmlparser.XMLWriter(odml_document).write_file(save_to)
```

Figure 3.4 – Example Python code to create an odML-file. (figure 8 of Zehl et al., 2016) The displayed Python code creates a small odML-file, which structure was used to introduce the structural design of the odML framework in Figure 1.2.

supported by the four cases illustrating why a comprehensive collection is useful: Case 2 and 3 demonstrate that the chosen metadata framework has to be human and machine-readable (reqt 1). Case 3 and 4 require that metadata within the chosen metadata framework are clearly defined and easy to call, as it would be via a key-value representation of the metadata (reqt 2). Case 1 illustrates the importance that the metadata framework should be flexible enough to include metadata of various types, to structure metadata to the specific needs on an experiment, and to enrich a metadata collection over time (reqt 3, 4, and 5). Case 4 determines that a metadata framework should be accessible via different programming languages, and, moreover, should make it possible to find and extract certain metadata, which can be linked with the used data framework to automatize and standardize a metadata guided data selection (reqt 6, 7, and 8). As already summarized in Section 1.2, the odML metadata framework, as introduced by Grewe et al. (2011), fulfilled already most of these requirements, and defective or missing aspects of the framework were optimized or added in collaboration with the original developers of the software and the current support of the project via the G-Node. Furthermore, I chose the odML framework especially because it is comparatively generic and flexible, which makes it well-suited for a broad variety of experimental scenarios and which enabled the SN-group to introduce it as metadata framework in multiple running collaborations.

This section complements the original technical publication of the software (Grewe et al., 2011) by illustrating in a tutorial-like style the preparations for creating a comprehensive metadata collection. As a practical example, I will use metadata of the described R2G experiment. In Figure 3.4 I display the Python code which creates a small exemplary odML-file, which structural design I introduced in Figure 1.2. In the next subsection, I will summarize the preparatory decisions that are needed to implement a complete odML metadata collection (Section 3.2.1 and Section 3.2.2) and outline a workflow how to enter and maintain metadata in such a collection (Section 3.2.3). In the next section (Section 3.3), I will complete the tutorial-like introduction of the odML framework by demonstrating the practical use of a corresponding metadata collection. Although I am convinced that the odML library is particularly well designed to manage metadata, the concepts and guidelines that I present are of general applicability, because most metadata frameworks are based on the same structural foundation, namely a hierarchical organization of metadata stored in key-value pairs (or triplets).

3.2.1 Distribution of information

In preparing a metadata management for an experiment, one has to decide if and how information should be distributed over multiple files. For example, one could decide to either generate (i) several files for each recording, (ii) a single file per recording, or (iii) a single file that comprises a series of recordings. The appropriate approach depends on both the complexity of metadata and the user's specific needs for accessing them. In the following, I will exemplify situations which could lead to one of the three different approaches:

(i) If the metadata of a preprocessing step are complex, combining them with the metadata related to the initial data acquisition could be confusing. One could instead generate separate files for the preprocessing and the recording. The downside of this approach is that the availability of both files needs to be assured.

(ii) In an experiment where each single recording comprises a certain behavioural condition, the amount of metadata describing each condition is complex, and recordings are performed independently from each other, a single comprehensive metadata file per recording should be used. For the R2G experiment, I chose this approach.

(iii) If one recording is strongly related to, or even directly influences, future recordings (e.g., learning of a certain behaviour in several training sessions), then one single comprehensive metadata file should cover all the related recordings. Even metadata of preprocessing steps could be attached to this single file.

3.2.2 Structuring information

Organizing metadata in a hierarchical structure facilitates navigation through possibly complex and extensive metadata files. The way to structure this hierarchy strongly depends on the experiment, the metadata content, and the individual demands resulting from how the metadata collection should be used. Nevertheless, there are some general guidelines to consider (based on Grewe et al. 2011):

guideline 1: Keep the structure as flat as possible and as deep as necessary. Avoid `Sections` without any `Properties`.

guideline 2: Try to keep the structure and content as generic as possible. This enables the reuse of parts of the structure for other recording situations or even different experiments. Design a common structure for the entire experiment, or even across related studies, so that the same set of metadata filters can be used as queries in upcoming analyses. If this is not possible and multiple structures are introduced, for instance, because of very different task designs, create a `Property` which one can use to determine which structure was used in a particular file (e.g., `Property` "*UsedTaskDesign*": `Value` "*2-inst*").

guideline 3: Create a structure that categorizes metadata clearly into different branches. Make use of the general components of an electrophysiological experiments (e.g. subject, setup, hardware, software, etc.), but also classify metadata according to context or time (e.g. previous knowledge, pre and post-processing steps).

guideline 4: In order to describe repeating entities like an experimental trial or an electrode description, it is advisable to separate constant `Properties` from those that change individually. Generate one `Section` for constant features and unique `Sections` for each repetition for metadata that may change.

In the following I will illustrate these principles with a subset of the metadata of the R2G experiment. Using the nomenclature of the odML metadata framework (cf. Section 1.2), I structured the metadata of an Utah-Array into the hierarchy of `Sections` which is schematically displayed in Figure 3.5.

The top level is called "*UtahArray*". As described in Section 2.2.2, a Utah-Array is a silicon-based multi-electrode array which is wired to a connector. General metadata of the Utah-Array, such as the serial number (cf. "*SerialNumber*" in Figure 3.5) are directly attached as `Properties` to the main `Section` (guideline 1).

More detailed descriptions are needed for the other components of the Utah-Array. The hierarchy is thus extended with a `Section` for the actual electrode array and a `Section` for the connector component. For both Utah-Array components, there are different fabrication types available which differ only slightly in their metadata configuration. For this reason, I named the `Sections` generically "*Array*" and "*Connector*" (guideline 2) and specified their actual fabrication type via their attached `Properties` (e.g., "*Style*").

The fabrication type of the "*Array*" is defined by the number and configuration of electrodes. In the R2G experiment a Utah-Array with 100 electrodes (96 connected, 4 inactive) was used, arranged on a 10x10 grid, supplied with wires for two references and one ground. It is, however, possible to have

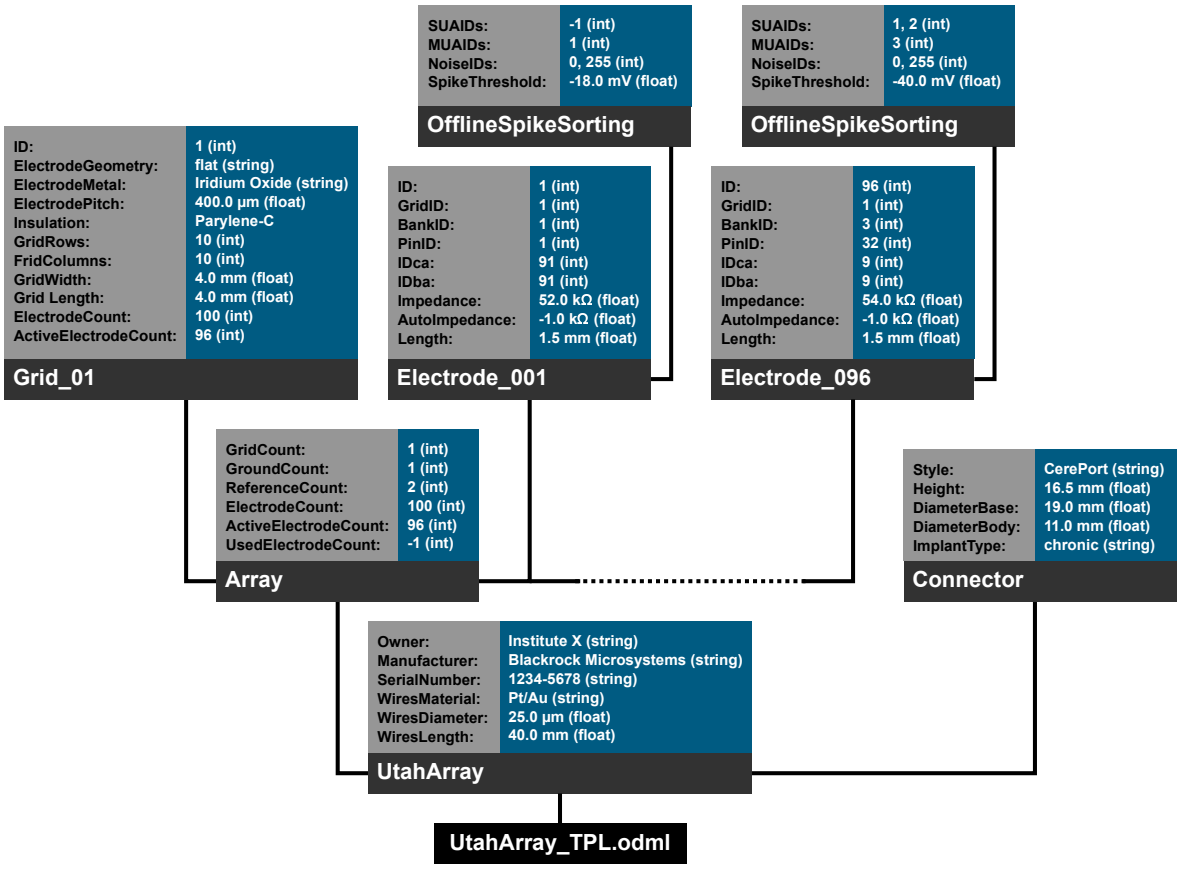

Figure 3.5 – Schematic view of the odML structure for metadata of a Utah-Array. (figure 6 of Zehl et al., 2016) Colour code of boxes matches colour code of odML objects in Figure 1.2. To simplify the schema, odML `Properties` and their `Values` are listed in blocks for each odML `Section` and additional attributes of the odML objects (e.g. definitions) are not displayed. Likewise, the remaining 94 odML `Sections` of the Utah-Array electrodes ("*Electrode_002*" to "*Electrode_095*") are left out and only indicated by the dashed line. Based on the size of the odML structure for a Utah-Array, which is only one branch of the structure of a final odML-file for the R2G experiment, one can imagine the complexity of a complete reach-to-grasp comprehensive metadata file.

a different total number of electrodes and even an array split into several grids with different electrode arrangements. Nevertheless, all electrodes or grids can be defined via a fixed set of `Properties` that describe the individual setting of each electrode or grid. To keep the structure as generic as possible, I registered, besides the total number of electrodes references and grounds, the number of active electrodes, and the number of grids as `Properties` of the (level-2-) Section "*Array*" (guideline 2). Within the "*Array*" `Section`, (level-3-) `Sections` named "*Electrode_XXX*" for each electrode and grid are attached, each containing the same `Properties`, but with individual `Values` for that particular electrode. This design makes it possible to maintain the structure for other experiments where the number and arrangement of electrodes and grids might be different (guidelines 2 and 4). The electrode IDs of a Utah-Array are numbered consecutively, independent of the number of grids. In order not to further increase the hierarchy depth, a `Property` "*Grid_ID*" in each "*Electrode_XXX*" `Section` identifies to which grid the electrode belongs to, instead of attaching the electrodes as (level-4-) `Sections` to its "*Grid_XX*" and "*Array*" parent `Sections` (guideline 1).

The example of the Utah-Array demonstrates the advantage of a meaningful naming scheme for `Sections` and `Properties`. To prevent ambiguity, any `Section` and `Property` name at the same level of the hierarchy must be unique. However, a given name can be reused at different hierarchy depths. This reuse can facilitate the readability of the structure and make its interpretation more intuitive. The "*UtahArray*" `Section` (Figure 3.5) demonstrates a situation in which ambiguous `Section` and `Property` names are useful, and where they must be avoided. The `Sections` for the individual electrodes of the array are all on the same hierarchy level. For this reason, their names need to be unique which is guaranteed by including the electrode ID into the `Section` name (e.g., "*Electrode_001*"). In contrast, the `Property` names of each individual electrode `Section`, such as "*ID*", "*GridID*", etc., can be reused. Similarly, the results of the offline spike-sorting can be stored in a (level-4-) `Section` "*OfflineSpikeSorting*" below each electrode `Section`. Using the identical name for this `Section` for each electrode is helpful, because its content identifies the same type of information. In Section 3.3, I show for the odML framework hands-on how one can make use of recurring `Section` or `Property` names to quickly extract metadata from large and complex hierarchies.

3.2.3 Workflow

For most experiments it is unavoidable that metadata are distributed across various files and formats (Section 2.5.1). To combine them into one or multiple file(s) of one standard format, one not only needs to generate a meaningful structure for organizing the metadata of the experiment, but also to write routines that load and integrate metadata into the corresponding file(s). For reasons of clarity and comprehensibility, I argue that these processes should be separated. Figure 3.6 illustrates and summarizes the corresponding workflow for the R2G experiment using the odML library.

As a first step, it is best to write a template in order to develop and maintain the structure of the files of a comprehensive metadata collection. In the odML metadata model, a template is defined as an "empty" odML-file in which the user determines the structure and attributes of `Sections` and `Properties` to organize the metadata, but fills it with dummy `Values`. To create an odML template one can use (i) the odML Editor, a graphical user interface that is part of the odML Python library (cf. Figure 3.10), (ii) a custom-written program based on the odML library (cf. Figure 3.4), or (iii) a spreadsheet software (cf. Figure 3.7). Especially for large and complex structures, (ii) or (iii) are the more flexible approaches to generate a template. To further simplify the editing process of templates, it is advisable to create multiple smaller templates (e.g. separated into the top-level `Sections`). These parts can then be handled independently and later easily merged back into a final structure.

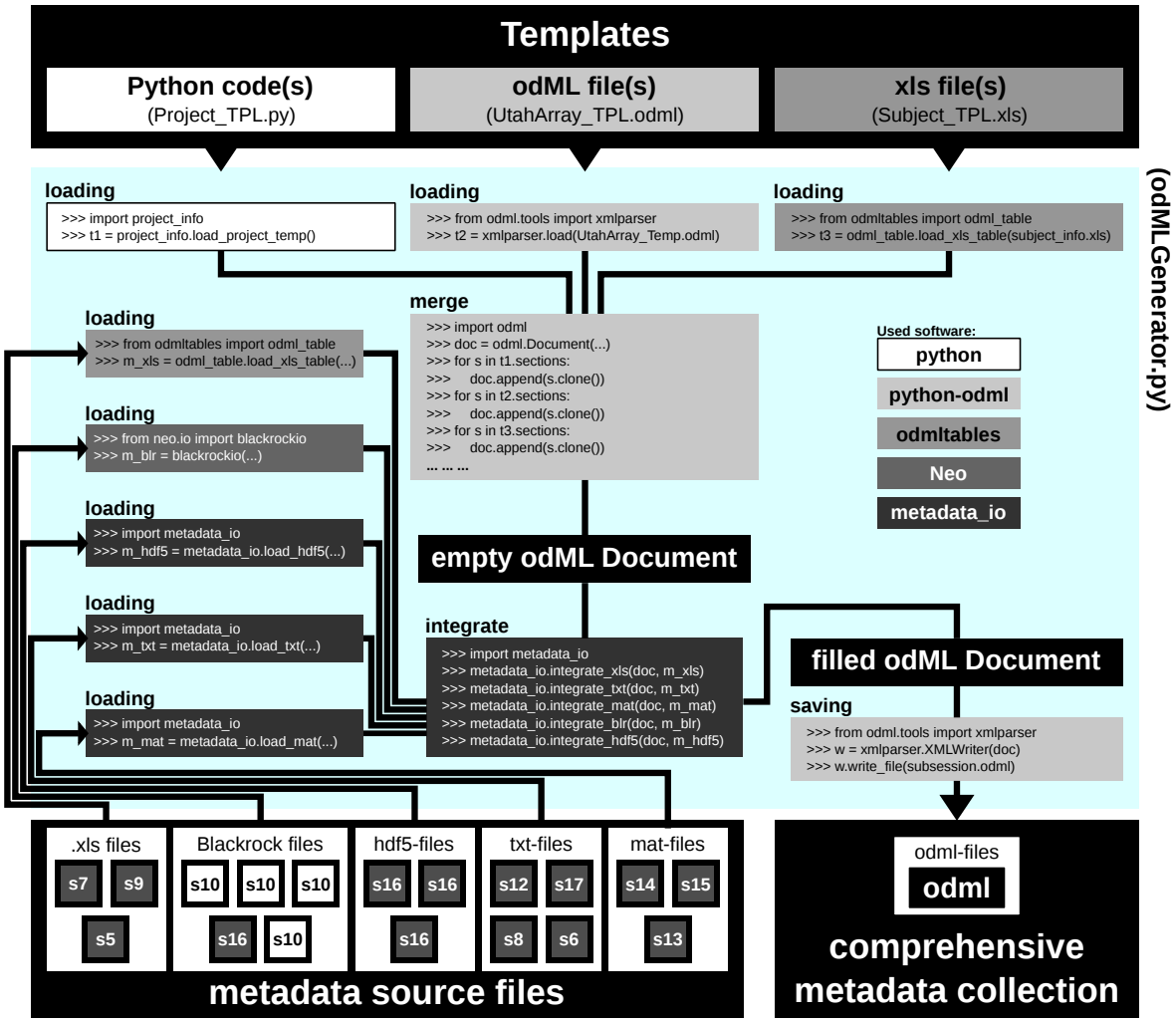

Figure 3.6 – Schematic workflow for generating odML-files in the R2G experiment. (modified figure 7 of Zehl et al., 2016) The *Templates* box (top): illustrates three out of six template parts which are used to build the complete, but (mostly) empty odML Document combining metadata of one recording in the R2G experiment. The three possible template formats (script, odML-file, and spreadsheet) are indicated within the grey shaded boxes (left to right, respectively), although I wrote the templates of the R2G experiment mainly as Python scripts. The *metadata source files* box (bottom left) illustrates all metadata sources of one recording (small labelled boxes, for labels cf. Section 2.5.1 and Figure 2.10) ordered according to file formats (white boxes). The central large cyan box shows the workflow of the odML generator routine (odMLGenerator.py). Code snippets used at the different steps of this generator are illustrated in the smaller white or grey scaled boxes whose shading represent the software used for the code (see legend right of centre). The black coloured boxes indicate files or file stages. The workflow consists of 5 steps: (i) loading all template parts, (ii) merging all template parts into one empty odML Document, (iii) loading all metadata sources of a recording, partially with custom-made routines (metadata_io), (iv) integrating all metadata sources into the empty odML Document using custom-made routines (metadata_io), and (v) saving the filled odML Document as odML-file of the corresponding recording. The *comprehensive metadata collection* box (bottom right) illustrates the reduction of all metadata sources of one recording to one odML-file.

This approach facilitates the development and editing even of large odML structures. The top of Figure 3.6 illustrates how three example template files (`Project_TPL.py`, `UtahArray_TPL.odml`, and `Subject_TPL.xls`) of the three possible template formats (i-iii, repectively) are merged into one large "empty" odML-file via the Python odML library.

For creating an odML spreadsheet template (cf. iii above) I would like to briefly introduce the odMLtables Python package[1]. I first implemented this framework providing only functions that converted odML-files into CSV or XLS spreadsheets and vice versa. The idea was to provide an odML-file in a familiar format often used in experimental laboratories, which can be used by every lab member to capture metadata in a standardized way (cf. Figure 3.8). In collaboration with J. Pick, a Bachelor student I supervised, and J. Sprenger, odMLtables was further formalized and extended by a GUI in form of a setup assistant (wizard) guiding through the functionality of the framework. The currently available version of odMLtables on GitHub[1] contains the following functionalities:

Conversion between odML-file and CSV or XLS spreadsheet. As mentioned, the core functionality of odMLtables is to convert odML-files into CSV or XLS spreadsheets and vice versa. The corresponding spreadsheets are structured as the odML-XLS template in Figure 3.7, although in a complete conversion the first row of the spreadsheet would list the `Document` attributes (`author`, `date`, `repository`, and `version`), and the table would be complemented by additional columns containing further attributes of the odML objects (`Data Uncertainty`, `Section Name`, `Section Type`, and `Section Definition`). The user can define whether the `Document` information row should be displayed, and which attribute column should be shown in which order in the the spreadsheet representation of the odML-file can be defined by the user. To be able to transfer a spreadsheet back into an odML-file, the provision of column `Path to Section`, `Property Name`, `Value`, and `odML Data Type` are mandatory. A useful gimmick of odMLtables for the XLS representation of an odML-file is that the user can define the colour scheme and font style of the spreadsheet. It is even possible to emphasize dummy `Values` to increase clarity and facilitate the editing process.

Filtering an odML-file. For a complex experiment an odML-file quickly becomes large and complex. To allow the user to convert only parts of a large odML structure, odMLtables provides filter functions, which are based on the `iter` and `filter`-functions of the odML library that will be introduced in Section 3.3.2. With this functionality the user is able to generate, for example, a XLS spreadsheet that only contains odML metadata entries that need to be manually edited.

Merging odML-files. odMLtables provides the functionality to merge two odML-files. This gives the user, for example, the possibility to create several odML-XLS spreadsheets of manageable size, that can then be merged into a larger final odML-file.

Comparing odML metadata entries. Within an odML structure, sometimes reoccurring Section structures are used to capture metadata of repeating entities like experimental trials or electrodes descriptions (cf. guideline 4 in Section 3.2.2). To gain an overview of such entities, odMLtables provides therefore the useful functionality to display them in a table for comparison. Note that the corresponding spreadsheets are not non-convertible into an odML-file.

The odMLtables framework is a running open-source project and further functionalities, such as comparing metadata entries across odML-files, are currently in development. A publication of the

[1] https://github.com/INM-6/python-odmltables

Path to Section	Property Name	Value	Data Unit	odML Data Type	Property Definition
/Subject	Species	-		string	Binomial species name (genus, species within genus)
	TrivialName	-		string	Commonly used (trivial) species name
	GivenName	-		string	Given name
	FileIdentifier	-		string	Identifier used in file names
	Gender	-		string	Gender (male or female)
	Birthday	11-11-1111		date	Date of birth (yyyy-mm-dd)
	ActiveHand	-		string	Trained hand (left and/or right)
	Disabilities	-		text	Comment on existing disabilities
	Character	-		text	Comment on the general character/personality
/Subject/ArrayImplant	Date	11-11-1111		date	Date of the surgery
	Surgeon	-		string	Full name(s) of the person(s) responsible for the surgery
	BodyWeight	-1.1	kg	float	Body weight of subject (pre-surgical)
	Hemisphere	-		string	Hemisphere (left and/or right) in which surgery was performed
	ArrayRotation	-1	deg	float	Rotation of the array (from connector aligned orientation) when implanted.
	Comment	-		string	Comment on the surgery
/Subject/Training	Coach	-		string	Full name(s) of person(s) responsible for the training
	BodyWeight	-1.1	kg	float	Body weight of subject (pre-training)
	Start	11-11-1111		date	Start date of training
	End	11-11-1111		date	End date of training
	Comment	-		string	Comment on training

Figure 3.7 – Example of an odML-XLS template. (modified figure 9 of Zehl et al., 2016) The displayed spreadsheet is a screen shot of the odML-XLS template used to collect information on each monkey in the R2G experiment (cf. Section 2.2 and source 6 in Section 2.5.1). The structure of the spreadsheet makes it possible to automatically transfer it to the odML format via the odMLtables package. The dummy `Values` (marked in cyan and blue) can be manually changed via, e.g., Excel, to the actual metadata `Values` for each monkey. In general, all columns can be manually edited and further columns added to provide additionally, e.g., a definition for each odML `Section`.

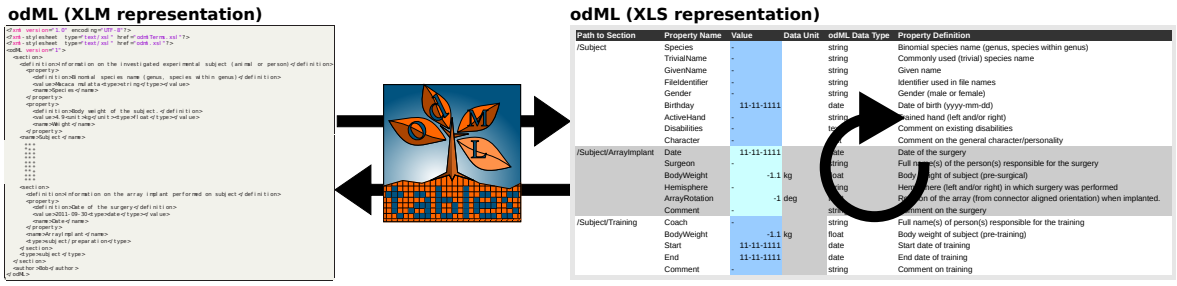

Figure 3.8 – Workflow of odMLtables. (modified figure 2 of the unpublished manuscript for Sprenger et al., prep) The schema summarizes the core functionality of odMLtables (indicated by its logo), a Python package to convert an odML-file (here displayed as XML-representation, cf. Figure 3.9) into a spreadsheet format (here exemplary the XLS representation, cf. Figure 3.7) and vice versa (indicated by straight arrows). The odMLtables library also includes a GUI in form of a wizard that guides the user through the functionalities of the conversion process. Within the spreadsheet, the user has the possibility to edit the content and the structure of the odML metadata representation (indicated by looping arrow). With this, the odML framework for collecting and storing metadata is also accessible by users with little or no programming knowledge.

software in a peer-reviewed journal is currently in preparation under the manuscript title "odMLtables: A user-friendly approach to managing metadata of neurophysiological experiments".

Let me now return to describe a workflow for creating an odML metadata collection. Once an odML template is created, in a second step, the user has to write or reuse a set of routines, which reads metadata from the various sources, integrates them into a copy of the template and saves them finally as a file of the comprehensive metadata collection (cf. Figure 3.6). Although it would be desirable if the complete workflow of loading and integrating metadata into one or multiple file(s) of a comprehensive metadata collection were automatized, in most experiments it cannot be avoided that some metadata need to be entered by hand (see Section 2.5.1). Importantly, I avoided direct manual modification of the comprehensive metadata files. Instead, the final comprehensive metadata collection of the R2G experiment is always created from the separately stored source files in which some metadata can be entered manually. This approach has the advantage that, if at one point the structure of the comprehensive metadata files needs to be changed, the manual entries will remain intact in the separately stored source files. Practically, manual metadata entries should be collected into source files with machine-readable formats, such as text files, CSV format, Excel, HDF5[2], JSON[3], or directly in the format of the chosen metadata management software (e.g., odML). If available, one can use the corresponding libraries of these formats to access and edit the source files, or to load and integrate the contained metadata into the file(s) of the comprehensive metadata collection. In the special case that these manual metadata entries are constant across the collection (e.g., project information, cf. s5 Section 2.5.1), one can further reduce the number of source files by entering the corresponding metadata directly into a template (e.g., Project_Temp in Figure 3.6). This avoids unnecessary clutter in the later compilation of metadata and facilitates consistency across odML-files. Metadata that are not constant across odML-files should be preferably stored in source files that adhere to standard file formats which can be loaded via generic routines. For this reason, all spreadsheet source files of the R2G experiment (cf. s5, s7, and s9 in Section 2.5.1 and Figure 2.10) were designed to be compatible with the odMLtables package (cf. example odML-XLS template in Figure 3.7). Furthermore, the files generated by the Blackrock DAQ system are loadable via the previously introduced, generic `BlackrockIO` of the Neo Python library (Garcia et al., 2014) or via the experiment-specific `ReachGraspIO`. Nevertheless, for the R2G experiment it was still necessary to write also custom loading and integration routines to cover all metadata from the various sources (e.g., txt, hdf5 and mat-files for results of the various preprocessing steps, cf. s Figure 3.6).

In summary, this two stage workflow of, first, generating templates, and, second, filling these templates with metadata from the multiple source files, guarantees flexibility and consistency of the comprehensive metadata collection over time. In particular, in a situation where the structure needs to be changed at a later time (e.g., if new metadata sources need to be integrated) one only needs to adapt or extend the template as well as the code that fills the template with metadata, in order to generate a consistent, updated comprehensive collection from scratch. Nonetheless, if the metadata management can be planned in advance, one should attempt to optimize the corresponding workflow in the following aspects:

- Use existing templates (e.g., the Utah-Array odML template) to increase consistency with other experimental studies.
- Keep the number of metadata sources at a minimum.

[2] https://www.hdfgroup.org

[3] http://www.json.org

- Avoid hidden knowledge in the form of handwritten notes or implicit knowledge of the experimenter by transferring such information into a machine-readable format (e.g., standardized Excel sheets compatible with odMLtables) early on.
- Automatize the saving of metadata as much as possible.

3.3 Using an odML metadata collection

In this section, I will now complement this chapter with practical demonstrations. Note that the code presented can be written in a more compact way, but for better readability I provide a longer, more explicit code format.

3.3.1 Manual inspection

As summarized in case 2 in Section 3.1, it can be quite useful to be able to manually inspect a metadata collection to get familiar with an experiment. As already mentioned in Section 1.2, there are three ways of manually screening an odML-file:

XML representation of an odML-file. The first possibility to open an odML-file would be to use a simple text editor (see Figure 3.9). This is possible, because odML is based on XML, which is a textual data format readable and editable with any available text editor. It is therefore a quick way to manually inspect or edit the content of an odML-file. However, the XML based representation is not convenient to edit for users, especially not for large, more complex structured odML-files.

Browser view of an odML-file. A second possibility to view, but not edit an odML-file is to open it via a web browser (see Figure 3.11). For this, one has to add an XSL-stylesheet (odML.xsl) to the directory where the odML-files are located before opening them to view. The XSL-stylesheet is available for download on the odML website[4]. The schema translates the XML based representation of odML into HTML code which is then interpreted by the web browser into an interactive web page representation. The web page will show, besides general information about the odML `Document`, the tree structure of the `Sections` as a static table of contents, and for each individual `Section` the corresponding `Properties` and `Values` as a flat table. Each `Section` entry in the static table of contents is a link to its corresponding flat content representation of `Properties` and `Values`. In turn, each content representation of `Properties` and `Values` has a back link (`top`) to the corresponding `Section` entry in the static table of contents. This approach is very useful for screening and browsing through an odML-file, especially if it is large and complex.

View of an odML-file via the odML-Editor. The third possibility to manually inspect or edit an odML-file is to use the odML-Editor (see Figure 3.10). The editor (`odml-gui`) is part of the Python odML library. Similar to the HTML view of an odML-file, the representation of the tree structure of the `Sections` is also separated from the flat representation of its `Properties` and `Values`. The editor window is subdivided into three parts. (1) The **`Sections`** pane displays a tree view of all `Sections` starting from the top level of the `Document`. (2) The **`Properties`** pane displays a table containing the `Name`, `Value` and the `Value` attributes (e.g., `Type`, `Unit`) of each `Property` belonging to a selected `Section`. (3) The **`Attributes`** pane displays the attributes of the current selected `Section`, `Property`

[4] http://www.g-node.org/projects/odml/tools

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet  type="text/xsl" href="odmlTerms.xsl"?>
<?xml-stylesheet  type="text/xsl" href="odml.xsl"?>
<odML version="1">
  <section>
    <definition>Information on the investigated experimental subject (animal or person)</definition>
      <property>
        <definition>Binomial species name (genus, species within genus)</definition>
        <value>Macaca mulatta<type>string</type></value>
        <name>Species</name>
      </property>
      <property>
        <definition>Body weight of the subject.</definition>
        <value>4.9<unit>kg</unit><type>float</type></value>
        <name>Weight</name>
      </property>
    <name>Subject</name>
    <section>
      <definition>Information on the array implant performed on subject</definition>
      <property>
        <definition>Date of the surgery</definition>
        <value>2011-09-30<type>date</type></value>
        <name>Date</name>
      </property>
      <name>ArrayImplant</name>
      <type>subject/preparation</type>
    </section>
    <type>subject</type>
  </section>
  <author>Bob</author>
</odML>
```

Figure 3.9 – XML representation of an odML-file. (based on listing S-1 of supplementary material of Zehl et al., 2016) XML-based representation of the small example odML-file introduced schematically in Figure 1.2 and as Python code in Figure 3.4. The XML code representation of an odML-file is editable with any available text editor, but due to the confusing syntax not convenient to work with.

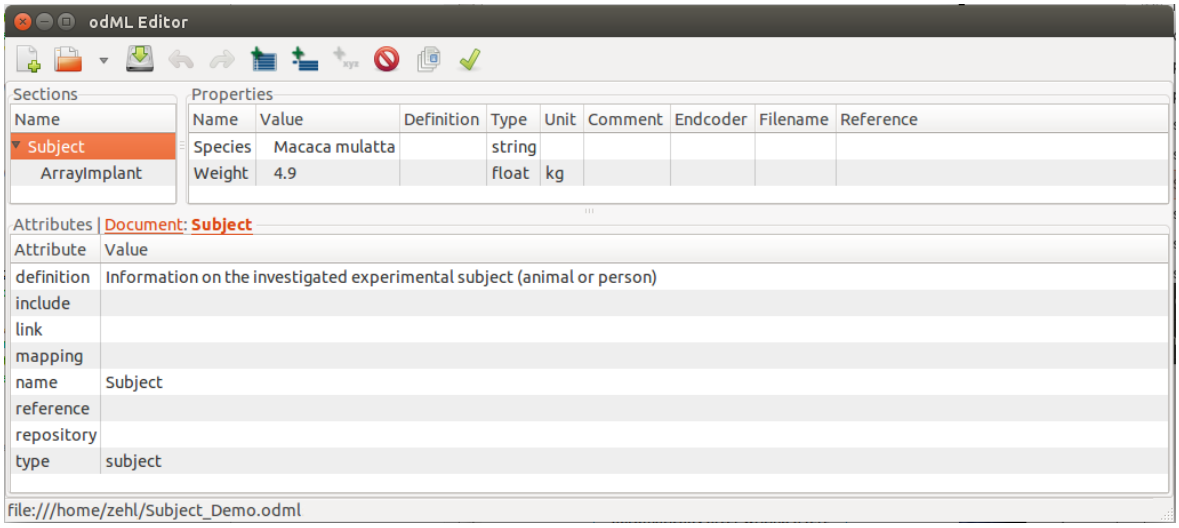

Figure 3.10 – View of an odML-file via the odML-Editor. (figure S-2 of supplementary material of Zehl et al., 2016) The displayed odML `Document` is equivalent to the schematic display in Figure 1.2, the corresponding Python implementation in Figure 3.4, and the XML-based representation in Figure 3.9. Note that in the `Sections` pane the "*`Subject`*" `Section` was selected (marked in orange in the sections pane). The corresponding `Properties` ("*`Species`*" and "*`Weight`*") are displayed in the `Properties` pane. The structure as well as all object entries (including `Section`, `Property` or `Document` attributes listed in the `Attributes` pane) are editable.

odML - Metadata

Document info

Author: Bob
Date:
Version:
Repository:

Structure

- Subject (type: subject)
  - ArrayImplant (type: subject/preparation)

Content

Section: Subject

Type: subject
Id:
Repository:
Link:
Include:
Definition: Information on the investigated experimental subject (animal or person)
Mapping:

Name	Value	Uncertainty	Unit	value id	Type	Comment	Definition
Species	Macaca mullata				string		Binomial species name (genus, species within genus)

top

Section: ArrayImplant

Type: subject/preparation
Id:
Repository:
Link:
Include:
Definition: Information on the array implant performed on subject
Mapping:

Name	Value	Uncertainty	Unit	value id	Type	Comment	Definition
Date	2011-09-30				date		Date of the surgery

top

Figure 3.11 – Browser view of an odML-file. (figure S-1 of supplementary material of Zehl et al., 2016) The displayed odML `Document` is equivalent to the schematic display in Figure 1.2, the corresponding Python implementation in Figure 3.4, and the XML-based representation in Figure 3.9. The `Document Info` pane summarizes the `Document` attributes. The `Structure` pane displays the `Sections` of an odML `Document` as static table of contents, which entries link to the corresponding `Sections` in the `Content` pane below. In the `Content` pane, for each `Section` the corresponding `Properties` and `Values` are displayed as flat table, with a back link (`top`) to the `Structure` pane after each `Section`. This approach is very useful for screening and browsing through a large and complex odML-file.

or `Document`. The header of the **`Attributes`** pane indicates the path to the selected `Section` or `Property` in red starting from the `Document` root.

3.3.2 Navigating the odML structure

Depending on the experiment, the odML structure can become large and complex, thus making it difficult to find certain metadata within this complex structure. For this reason and on my request, the odML Python library now provides helper functions which can be used to find and extract metadata with minimal user knowledge on the odML structure. In the following, I will demonstrate how these helper functions, `itervalues`, `iterproperties` and `itersections` (collectively referred to as iter-functions), can be used in the previously cases if a comprehensive odML metadata collection with one odML-file per recorded dataset was generated (cf. Section 3.1).

Extract unique objects from an odML file in Python. Let us assume that I wrote an analysis script to test if the firing rates depend on the behavioural condition in the trials, and I now want to run this analysis script now on single unit (SUA) data pooled across the datasets of the standard task paradigm (2-inst) with the condition setting 1 and random trial type sequence (cf. Section 2.1.1). Thus, I need to check which dataset was already offline spike sorted. From a previous manual inspection of the odML-files, I remember that the `Property` containing this information was called "*`IsSpikeSorted`*". I also remember that this `Property` name is unique and that the odML data type of the metadata `Value` saved in this `Property` is a boolean (`True` or `False`). I cannot remember, though, where this `Property` is located in the complete tree structure of the R2G odML-files (for an example on how to extract odML objects via their absolute path in the hierarchy, see the odML Python tutorial[5]). Nonetheless, my knowledge is sufficient enough to make use of the Python odML helper function `iterproperties`, which iterates through all `Property` objects of an odML-file, and combine it with a filter function that checks for each `Property` object if its name is equal to "*`IsSpikeSorted`*" (see Figure 3.12). As result, I receive a list containing exactly one `Property` containing the requested metadata. I extract the `Property` from the list and accesses the single `Value` object of the `Property` to extract the stored metadata of type boolean to print out if the dataset I looked at (e.g., l101126-002 in Figure 3.12) was offline spike sorted or not.

Extract ambiguous objects from an odML. If an odML `Property` or `Section` name is ambiguous, one can extend the filter function to check for several attributes of the requested object. For example, let me assume I want to know the SUA IDs of one particular electrode with the ID 11. Again from previous inspections of the odML-file, I remember that the `Property` name which contains the metadata I am searching for is "*`SUAIDs`*" and that it exists as a child object below each of 96 uniquely named `Sections` which represent the active electrodes of the Utah-Array (cf., Figure 3.5). I make use of this fact and extend the filter-function not only to make sure that the `Property` name is "*`SUAIDs`*", but also that the name of the `Section` of the particular electrode "*`Electrode_011`*" occurs in the path of the requested `Property` (see Listing S-3). I combine this more complex filter-function with the odML helper function `iterproperties` and extracts the requested `Property` from the resulting list. I am aware that the `Property` "*`SUAIDs`*" can contain multiple `Values` which I write into a list. I then loop through this list to access the `Values` containing the SUA IDs of the electrode with ID 11.

Which iter-function one has to use, and how complex the filter-function should be, depends on both the structure of the odML-file and the user's need. For automatic extraction of metadata one

[5] http://g-node.github.io/python-odml/

```python
import odml

# load an odML file of one recording
odml_filename = '1101126-002.odml'
odml_of_recording = odml.tools.xmlparser.load(odml_filename)

# define a filter function which checks if the name of object (x) equals
# "IsSpikeSorted"
def filter_function(x):
    return x.name=="IsSpikeSorted"

# combine iter and filter function to collect all Properties fullfilling the
# given condition
list_of_properties = odml_of_recording.iterproperties(filter_function)

# if you know the Property is unique, extract it from this list
wanted_property = list_of_properties[0]

# if you know it contains only a single Value, extract it from the wanted
# Property
value_of_wanted_property = wanted_property.value

# extract the metadata (here, boolean) from the Value
metadata = value_of_wanted_property.data


# remove extension from odml_filename
odml_filename_rmext = os.path.splitext(odml_filename)[0]
# get only filename of recording
recording_filename = os.path.basename(odml_filename_rmext)

# print out if recording was spike sorted
if metadata == True:
    print("recording %s was spike sorted" % recording_filename)
else:
    print("recording %s is NOT yet spike sorted" % recording_filename)
```

Figure 3.12 – Extract unique objects from an odML-file in Python. (listing S-2 of supplementary material of Zehl et al., 2016) Python code for extracting an odML object with a unique name. The example demonstrates how one makes use of a filter-function for a `Property` (named "*`IsSpikeSorted`*") in combination with the corresponding Python odML function `iterproperties`.

```python
import odml

# load an odML file of one recording
odml_filename = 'l101126-002.odml'
odml_of_recording = odml.tools.xmlparser.load(odml_filename)

# define a filter function which checks if the name of object (x) equals
# "SUAIDs" and the Section name "Electrode_011" occurs in the object's path
def filter_function(x):
    return x.name == "SUAIDs" and "Electrode_011" in x.get_path()

# combine iter and filter function to collect all Properties fullfilling the
# given condition
list_of_properties = odml_of_recording.iterproperties(filter_function)

# if you know the Property is unique, extract it from this list
wanted_property = list_of_properties[0]

# if you know it contains multiple Values, extract them as list from the
# wanted Property
list_of_values_of_wanted_property = wanted_property.values

# loop through the list of Values to access metadata and print out SUA IDs of
# Electrode_011
print("SUA IDs of Electrode_011:")
for value_of_wanted_property in list_of_values_of_wanted_property:
    metadata = value_of_wanted_property.data
    print(metadata)
```

Figure 3.13 – Extract ambiguous objects from an odML-file via search conditions in Python. (listing S-3 of supplementary material of Zehl et al., 2016) Python code for extracting an odML object with an ambiguous name by extending the search conditions given in the filter-function. The example demonstrates how one makes use of a filter-function for a `Property` (named "*SUAIDs*") that is the child of a specific `Section` (named "*Electrode_011*") the corresponding Python odML function `iterproperties`.

```python
import odml

# load an odML file of one recording
odml_filename = '1101126-002.odml'
odml_of_recording = odml.tools.xmlparser.load(odml_filename)

# define a filter function which checks if the name of object (x) equals
# "Electrode_011"
def filter_function_sec(x):
    return x.name == "Electrode_011"

# combine iter and filter function to collect all Sections fullfilling the
# given condition
list_of_sections = odml_of_recording.itersections(filter_function_sec)

# if you know the Section is unique, extract it from this list
wanted_section = list_of_sections[0]

# define a new filter function which checks if the name of object (x) equals
# "SUAIDs"
def filter_function_prop(x):
    return x.name == "SUAIDs"

# use iter function on the wanted Section only to filter all Properties
# fullfilling the given condition
list_of_properties = wanted_section.iterproperties(filter_function_prop)

# if you know the Property is now unique, extract it from this list
wanted_property = list_of_properties[0]

# if you know it contains multiple Values, extract them as list from the
# wanted Property
list_of_values_of_wanted_property = wanted_property.values

# loop through the list of Values to access metadata and print out SUA IDs of
# Electrode_011
print("SUA IDs of Electrode_011:")
for value_of_wanted_property in list_of_values_of_wanted_property:
    metadata = value_of_wanted_property.data
    print(metadata)
```

Figure 3.14 – Extract ambiguous objects from an odML-file via partial searches in Python. (listing S-4 of supplementary material of Zehl et al., 2016) Python code for extracting an odML object with an ambiguous name by narrowing down the search to a smaller part of the odML Document. The example demonstrates how to first make use of a filter-function for a `Section` (name "*`Electrode_011`*") in combination with the Python odML function `itersections` to extract the corresponding odML branch, and then use only on this branch a filter-function for a `Property` (named "*`SUAIDs`*") in combination with the corresponding Python odML function `iterproperties`.

```
import odml

# load an odML file of one recording
odml_filename = 'l101126-002.odml'
odml_of_recording = odml.tools.xmlparser.load(odml_filename)

# define a filter function which checks if the name of object (x) equals
# "SUAIDs"
def filter_function(x):
    return x.name == "SUAIDs"

# combine iter and filter function to collect all Properties fullfilling the
# given condition
list_of_properties = odml_of_recording.iterproperties(filter_function)

# make use of the list of all Properties named 'SUAIDs' to identify how many
# single units were identified on all electrodes
suaid_number = 0
for prop in list_of_properties:
    list_of_values = prop.values
    for val in list_of_values:
        metadata = val.data
        if metadata in range(1, 17):
            suaid_number += 1

# print how many SUA IDs were identified in this recording
print("Number of SUA IDs:", suaid_number)
```

Figure 3.15 – Extract all ambiguous objects from an odML-file in Python. (listing S-5 of supplementary material of Zehl et al., 2016) Python code for extracting a list of related odML objects with ambiguous names. The example demonstrates how one makes use of a filter-function to extract all ambiguously named `Property` objects (here, named "*SUAIDs*") to gain at once access to the metadata in the `Value` objects related to these `Properties`.

has to make sure that the filter function is complex enough to guarantee that the iter-function returns only the requested objects. In case of a large odML structure one should narrow the search down to a certain branch of the odML-file and avoid iterations through `Value` objects of the odML-file. Both will save run time in the implementation. The search for the `Property` "*SUAIDs*" of `Section` "*Electrode_011*", for example, could have been also written differently. In Figure 3.14, I assume to know that, although the `Property` name "*SUAIDs*" exists many times within the odML-file, the `Section` name "*Electrode_011*" is unique and below this `Section` only one `Property` is named "*SUAIDs*". I can make use of this fact by dividing the search for the requested SUA IDs in two steps. In step one, I create a filter-function in combination with the odML helper function `itersections` to find and extract the `Section` with name "*Electrode_011*" from the odML-file. In the second step, I use the odML helper function `iterproperties` not on the entire odML-file, but on the extracted `Section` "*Electrode_011*" in combination with a filter-function for finding the `Property` named "*SUAIDs*". Note that the fastest way to access metadata of an odML-file is via the absolute path to the corresponding `Value` object. For example, the code between line 6 and 15 in Figure 3.14 can be replaced by accessing the desired `Section` in line 17 via its absolute odML path[6][7] (cf. also the odML Python tutorial[8]):

To get an idea about the quality of the spike detection and the offline spike sorting, I can also make use of the ambiguity of the odML `Property` name "*SUAIDs*" to collect the number of SUA identified for all electrodes of the Utah-Array (cf. Figure 3.15 and example use case in Figure 2.9). To extract the SUA numbers for an overview figure such as Figure 2.9, I combine the odML helper function `iterproperties` with a filter-function that only checks if the odML `Property` name equals "*SUAIDs*" and count, based on the resulting lists of odML `Properties` with name "*SUAIDs*", how many odML `Values` contain metadata matching SUA IDs.

3.3.3 Navigating across odML-files

While in Section 3.3.2, I demonstrated how to locate and access specific metadata in the hierarchy within an odML-file, here I will illustrate how to apply this mechanism across odML-files. As a simple example, I want to create a list of file-names from datasets which were already offline spike sorted. For this, I define a corresponding metadata criterion, namely that the `Property` named "*IsSpikeSorted*" should contain the `Value` "*True*", and extract and test the criterion in each odML-file of the collection using the previously introduced iter and filter-functions (cf. Figure 3.16). If the criterion is fulfilled, I append the corresponding filename automatically to a list of offline spike sorted datasets.

If I have more than one criterion to be used for selecting datasets or even specific data from recordings (e.g., specific trials) it is helpful to combine all checks based on the `iterproperties` function in a single criterion function for increased clarity. For example, based on the code examples in Figure 3.12 and Figure 3.15, I may create a criterion function which checks if a datasets was offline spike sorted and if so, if the number of identified SUA was larger than 60 (Figure 3.17).

3.3.4 Integration of additional metadata

Additional technical validations of the data (e.g., offline spike sorting or quality assessment) as described in Section 2.4 are often performed over a time period of months after the actual recording which is typical for a workflow of an electrophysiological experiment. Nonetheless, the access to the

[6] `e.g., wanted_property = odml_of_recording.sections["UtahArray"]["Array"]["Electrode_011"]`

[7] `e.g., wanted_property = odml_of_recording.sections[4][1][11]`

[8] http://g-node.github.io/python-odml/

```python
import os
import odml

# define the directory to the odml files of all recordings
directory_of_odmlfiles = os.getcwd()

# get all odml filenames
list_odml_filenames = os.listdir(directory_of_odmlfiles)

# create an empty list for filenames of spike sorted recordings
list_spikesorted_filenames = []

# define a filter function which checks if the name of object (x) equals
# "IsSpikeSorted"
def filter_function(x):
    return x.name == "IsSpikeSorted"

# loop through the extracted list of odml filenames
for odml_filename in list_odml_filenames:
    # load odml file of recording
    odml_of_recording = odml.tools.xmlparser.load(odml_filename)

    # combine iter and filter function to collect all Properties
    # fullfilling the given condition
    list_of_properties = odml_of_recording.iterproperties(filter_function)

    # if you know the Property is unique, extract it from this list
    wanted_property = list_of_properties[0]

    # if you know it contains only a single Value, extract it from the
    # wanted Property
    value_of_wanted_property = wanted_property.value

    # extract the metadata (here, boolean) from the Value
    metadata = value_of_wanted_property.data

    # append to list if recording was spike sorted
    if metadata == True:
        # remove extension from odml_filename
        odml_filename_rmext = os.path.splitext(odml_filename)[0]
        # get only filename of recording
        recording_filename = os.path.basename(odml_filename_rmext)

        # append recording_filename to list of spike sorted recordings
        list_spikesorted_filenames.append(recording_filename)
```

Figure 3.16 – Extract unique objects from a set of odML-files for a data selection in Python. (listing S-6 of supplementary material of Zehl et al., 2016) Python code for extracting an odML object with a unique name from a set of odML-files to create a list of file-names of datasets fulfilling the requested metadata criterion (`Property` named "*`IsSpikeSorted`*" contains `Value` "*`True`*").

```python
import os
import odml

# define a filter function which checks if the name of object (x) equals
# "IsSpikeSorted"
def is_spikesorted(x):
    return x.name == "IsSpikeSorted"

# define a filter function which checks if the name of object (x) equals
# "SUAIDs"
def name_is_suaids(x):
    return x.name == "SUAIDs"

# define criteria function
def criteria_function(odml_of_recording):
    """Checks if a recording was spike sorted, and if yes, if the number
    of identified single units is larger than 60"""

    # combine iter and filter function to collect all Properties fullfilling
    # the given condition
    list_of_properties = odml_of_recording.iterproperties(is_spikesorted)

    # if you know the Property is unique, extract it from this list
    wanted_property = list_of_properties[0]

    # if you know it contains only a single Value, extract it from the wanted
    # Property
    value_of_wanted_property = wanted_property.value

    # extract the metadata (here, boolean) from the Value
    metadata = value_of_wanted_property.data

    if metadata == True:
        # combine iter and filter function to collect all Properties
        # fullfilling the given condition
        list_of_properties = odml_of_recording.iterproperties(name_is_suaids)

        # make use of the list of all Properties named 'SUAIDs' to identify
        # how many single units were identified on all electrodes
        suaid_number = 0
        for prop in list_of_properties:
            list_of_values = prop.values
            for val in list_of_values:
                metadata = val.data
                if metadata in range(1, 17):
                    suaid_number += 1
        if suaid_number > 60:
            return True
        else:
            return False
    else:
        return False

# define the directory to the odml files of all recordings
directory_of_odmlfiles = os.getcwd()

# get all odml filenames
list_odml_filenames = os.listdir(directory_of_odmlfiles)

# create an empty list for filenames of spike sorted recordings with a
# sufficient number of single units
list_filenames = []

# loop through the extracted list of odml filenames
for odml_filename in list_odml_filenames:
    # load odml file of recording
    odml_of_recording = odml.tools.xmlparser.load(odml_filename)

    # use the defined criteria function to test if recording was spike sorted
    if criteria_function(odml_of_recording) == True:
        # remove extension from odml_filename
        odml_filename_rmext = os.path.splitext(odml_filename)[0]
        # get only filename of recording
        recording_filename = os.path.basename(odml_filename_rmext)

        # append recording_filename to list of spike sorted recordings
        list_filenames.append(recording_filename)
```

Figure 3.17 – Extract multiple objects from an odML-file for a data selection in Python. (listing S-7 of supplementary material of Zehl et al., 2016) Python code that uses a criterion function to generate a list of file-names of offline spike sorted datasets, which contain at least 60 identified SUAs.

parameters and results of these technical validations in form of metadata, usually becomes highly significant for all researchers working on the data (cf. case 1 Section 3.1). Here, I will now illustrate different scenarios of how metadata of such delayed technical validations can be gradually integrated into an existing odML-file.

In a first scenario, the technical validation is expected and known in advance (e.g., offline spike sorting). Here, the odML structure can be planned ahead with default dummy metadata `Values` in the form of an odML template (cf. Section 3.2.3). In such a case, it is possible to replace the dummy `Values` in the odML structure by the upcoming actual metadata `Values` of the technical validation when they emerge.

Alternatively, the technical validation may not have been expected, for example, if its importance arises only after performing preliminary analyses of the data (e.g., gap correction). Indeed, the ideal odML structure for metadata may only become clear during development of a new technical validation procedure and needs to be integrated into the existing odML-files later on. In such a case, one should update the original odML structure, either completely or by integrating a with the a new branch, update the corresponding metadata integration routines (cf. Section 3.2.3), and rerun the generation of all odML-files to keep overall consistency and reproducibility.

3.3.5 odML access via Matlab

In this final subsection, I will discuss the situation where two scientists (e.g, Alice and Bob in Figure 3.2) from different laboratories decide to work together even though they use different programming languages for data analysis (e.g., MATLAB and Python, respectively). The question arises how both scientists can use odML without abandoning their preferred programming language. Indeed, odML libraries exist not only for Python but also for MATLAB and Java. As MATLAB is often used in experimental neurophysiological laboratories, I illustrate here how the previously stated Python code examples can be translated into MATLAB. The current version of the MATLAB odML library provides an API that differs from the Python-odML library. In particular the MATLAB API is limited to load data from odML-files but not to write to odML-files, and the flexible `iterproperties` is replaced by a comparable, but more limited, helper function called `odml_find`. When using the MATLAB odML library[9], the odML objects are stored in MATLAB structure arrays, making handling of the corresponding metadata convenient and familiar for MATLAB users. It must be noted that the `odml_load` loading function provides an option to choose between two possible ways of mapping the odML objects to the fields of the structure array. Using the default "**`tree`**" option, the fields of the structure array are directly named after the names of the `Sections` and `Properties` defined in the odML-file (cf. (cf. Python code to access a `Section` via the absolute path[10]), as illustrated in Figure 3.18. Using the "**`odml`**" option, the odML objects are loaded in the structure array following more closely the Python odML object model, but only by calling the objects via their position and not via their name (cf. Python code to access a `Section` via the absolute path[11]), as illustrated in Figure 3.19.

As can be inferred by comparing the code in Figure 3.18 and Figure 3.19, using one or the other option will be more suitable depending of the type of processing that will be performed on the metadata. For example, when the user knows the odML hierarchy and wants to directly access a given `Property`, the option "**`tree`**" leads to more explicit naming in the structure array fields, which

[9] https://github.com/G-Node/matlab-odml

[10] `e.g., wanted_property = odml_of_recording.sections["UtahArray"]["Array"]["Electrode_011"]`

[11] `e.g., wanted_property = odml_of_recording.sections[4][1][11]`

```
odml_config;

% load the test odML file with the default 'tree' option
rootSection = odml_load('example_odML.odml');

% access the value of the weight of the subject
weight = rootSection.Subject.Weight.value;
```

Figure 3.18 – Load an odML-file in MATLAB - "`tree`" option. (listing S-8 of supplementary material of Zehl et al., 2016) MATLAB code for accessing a specific metadata value (e.g., `Weight`) when using the default "`tree`" loading option.

```
odml_config;

% load the test odML file with the 'odml' option
rootSection = odml_load('example_odML.odml', 'odml');

% access the value of the weight of the subject
weight = rootSection.section(1).property(2).value(1).value;
```

Figure 3.19 – Load an odML-file in MATLAB - "`odml`" option. (listing S-9 of supplementary material of Zehl et al., 2016) MATLAB code for accessing a specific metadata value (e.g., `Weight`) when using the default "`odml`" loading option.

```
odml_config;

% load an odML file of one recording
odml_filename = 'l101126-002.odml';
odml_of_recording = odml_load(odml_filename);

% if you know that the Property name 'IsSpikeSorted' is unique, extract the
% corresponding Property
wanted_property = odml_find(odml_of_recording, 'IsSpikeSorted');

% extract the metadata (here, boolean) from the Property, knowing it
% contains a single value
metadata = wanted_property.value;

% printout if recording was spike sorted
if metadata == true
    disp('recording was spike sorted');
else
    disp('recording is NOT yet spike sorted');
```

Figure 3.20 – Extract unique objects from an odML file in MATLAB. (listing S-10 of supplementary material of Zehl et al., 2016) MATLAB code for extracting an odML object with a unique name (cf. with Python code in Figure 3.12).

```
odml_config;

% load an odML file of one recording
odml_filename = '1101126-002.odml';
odml_of_recording = odml_load(odml_filename, 'odml');

% if you know that the Section name 'Electrode_011' is unique, extract the
% corresponding Section
wanted_section = odml_find(odml_of_recording, 'Electrode_011');

% if you know that the Property name 'SUAIDs' is unique in the wanted
% Section, extract the corresponding Property
wanted_property = odml_find(wanted_section, 'SUAIDs');

% if you know it contains multiple Values, extract them as struct from the
% wanted Property
struct_of_values_of_wanted_property = wanted_property.value;

% loop through the struct of Values to access metadata and printout SUA IDs
% of Electrode_011
disp('SUA IDs of Electrode_011:');
for ind = 1:length(struct_of_values_of_wanted_property)
    value_of_wanted_property = struct_of_values_of_wanted_property(ind);
    metadata = value_of_wanted_property.value;
    disp(metadata);
end
```

Figure 3.21 – Extract ambiguous objects from an odML file in MATLAB. (listing S-11 of supplementary material of Zehl et al., 2016) MATLAB code for extracting an odML object with an ambiguous name (cf. with Python code in Figure 3.14).

```
odml_config;

% load an odML file of one recording
odml_filename = '1101126-002.odml';
odml_of_recording = odml_load(odml_filename, 'odml');

% extract all the Properties having the name 'SUAIDs'
list_of_properties = odml_find(odml_of_recording, 'SUAIDs', Inf);

% make use of the list of all Properties named 'SUAIDs' to identify how
% many single units were identified on all electrodes
suaid_number = 0;
for ind_prop = 1 :length(list_of_properties)
    prop = list_of_properties{ind_prop};
    struct_of_values = prop.value;
    for ind_val = 1:length(struct_of_values)
        val = struct_of_values(ind_val);
        metadata = val.value;
        if metadata >= 1 && metadata <= 17
            suaid_number = suaid_number + 1;
        end
    end
end

% print how many SUAIDs were identified in this recording
disp('Number of SUAIDs:');
disp(suaid_number);
```

Figure 3.22 – Extract all ambiguous objects from an odML file in MATLAB. (listing S-12 of supplementary material of Zehl et al., 2016) MATLAB code for extracting a list of related odML objects with ambiguous names (cf. with Python code in Figure 3.15).

makes writing and reading of the code more convenient. For some more advanced processing, in particular when looping over metadata structures, or when the number of `Values` or `Properties` or `Sections` is unknown, the "`odml`" option can be more suitable. A more detailed discussion about the two loading options can be found in the help of the `odml_load` function.

The `odml_find` function searches an odML structure for the `Section` or `Property` object with a given name or all `Section` and `Property` objects with the given name. It can be used to build the MATLAB code equivalent to the Python code that I presented in the previous subsections. In the simple case where the requested object is uniquely represented in the odML-file, as it is the case in the code example, in which I wanted to find out if a particular dataset was already offline spike sorted (cf. Figure 3.12), the MATLAB code is straight-forward (see Figure 3.20).

In the more complex situation where the requested object is not uniquely represented in the odML-file, as in the code example in which I wanted to know the SUA IDs of the electrode with ID 11 (cf. Figure 3.13), I first need to extract the unique identifiable electrode `Section` "*Electrode_011*" and then use the resulting `Section` object in the `odml_find` function to access the metadata of the `Property` named "*SUAIDs*" (see Figure 3.21).

This approach is not as flexible as the Python code using the combination of `iterproperties` with complex filter-functions as demonstrated in Figure 3.13, but it is very similar to the approach in Figure 3.14 and still powerful enough to access any metadata within the odML-file with little detailed knowledge of the odML-file structure. Finally, as illustrated in Figure 3.22, I can also make use of the `odml_find` function to find a list of `Properties` with ambiguous names. For example, I may wish to collect the number of SUAs identified for all electrodes of the Utah-Array, as I did in the Python code example in Figure 3.15.

Chapter 4

Publishing research of the example experiment

The following chapter is based in main parts on the text of the manuscript "LFP beta amplitude is predictive of the mesoscopic spatio-temporal phase patterns". The manuscript will be submitted to Neuron. Preliminary results of this research were presented in abstract form in (Denker et al., 2014b, 2015b).

In the following chapter, I will present the practical usage of the data descriptor of the R2G experiment, as well as the concepts and frameworks for data and metadata management I introduced in Chapter 2 and Chapter 3 in form of an exemplary research analysis on selected datasets of the R2G experiment. The manuscript presented here describes the methods and results of this analysis and will be published under the title "LFP beta amplitude is predictive of the mesoscopic spatio-temporal phase patterns". This work was a highly collaborative research project between scientists of the SN and CoMCo groups (cf. authors contributions listed under "List of own papers with authors contributions"). For this reason, I will switch for this chapter from the first-person to the we perspective.

The analysis focuses on oscillations of recorded local field potentials (LFP) in the beta ($15-35 Hz$) range, that are one of the most prominent physiological correlates of behaviour observed in motor cortex. On the level of the LFP, beta oscillations recorded on separate electrodes in motor cortex are often highly correlated, but exhibit a non-zero phase shift. These shifts have been shown to organize spatially in the form of planar wave propagation along preferred directions across the cortical surface during an instructed-delay reaching task. However, little has been reported about the spatial organization of beta oscillations outside epochs that exhibit a clear planar wave. We demonstrate for the R2G experiment that a variety of additional spatial patterns of LFP beta activity may be distinguished outside epochs that exhibit planar waves. Moreover, we demonstrate that the trial-by-trial modulation of beta power in spindles directly relates to the observed wave pattern. Based on the instantaneous phase and phase gradients of the beta-filtered LFP, we introduce and combine measures that enable to identify different spatial activity patterns: (i) planar waves, (ii) (quasi) synchronized states (all electrodes appear synchronized at near-zero lag), (iii) spatially random states, (iv) inward or outward pointing radial patterns, and (v) circular patterns. We assess the behavioural relationship and statistical properties of the patterns, including their duration and average direction. In particular, we relate the observed patterns to beta spindles identified by large instantaneous amplitudes. We find that the pattern of wave propagation correlates with the beta power, where the peak of spindles

typically coincides with a quasi-synchronized state (ii), flanked by periods of planar (i) and radial (iv) wave patterns. In combination with previous observations, we argue that this result supports the hypothesis that beta power is indicative of spatio-temporal organization of spike synchronization.

4.1 Background of the scientific question

Before we go into details of the methods and results of the conducted analysis, we first want to summarize the scientific background which led us to address the scientific question how LFP beta oscillations spatially organize in the motor cortex, and if their spatial organization can be used to interpret the underlying network activity.

The LFP has long served as a readily available brain signal to monitor the average input activity that reaches the neurons in the vicinity of extracellular recording electrodes (Mitzdorf, 1985; Logothetis and Wandell, 2004; Einevoll et al., 2013). A hallmark of the LFP is the ubiquitous presence of oscillations in various frequency bands (Buzsáki and Draguhn, 2004) modulating in time and in different brain structures. These oscillations have been linked to a variety of brain processes such as attention (Fries et al., 2001), stimulus encoding (Engel et al., 1990), or memory formation (Pesaran et al., 2002; Dotson et al., 2014), and form the basis of modern theories concerning the functional implication of oscillatory brain activities, such as feature binding (Singer, 1999), the concept of communication-through-coherence (Fries, 2005; Womelsdorf et al., 2007; Fries, 2015), the phase-of-firing coding (Masquelier et al., 2009), or predictive coding (Friston et al., 2015). In motor cortex, beta oscillations (in the range of $15-35 Hz$) are one of the most prominent types of oscillatory activity. They have been linked to states of general arousal, movement preparation, or postural maintenance (Kilavik et al., 2012, for review see Kilavik et al., 2013), and are typically suppressed during active movement (cf. Pfurtscheller and Aranibar, 1979; Rougeul et al., 1979).

Recent progress in recording technologies led to the development of multi-electrode arrays allowing to record massively parallel neuronal signals in a precisely identifiable spatial arrangement. Although LFP signals recorded in motor cortex from electrodes separated by up to several millimetres are typically highly correlated (Murthy and Fetz, 1996a), the analysis of the instantaneous phase of the oscillation (e.g., Varela et al., 2001) revealed a non-zero temporal shift between electrodes (Denker et al., 2011). Such shifts may be expressed by dynamic spatial pattern formations propagating along preferred directions across the cortical surface, referred to as travelling waves (Rubino et al., 2006). Indeed, the phenomenon of travelling waves has been described in multiple brain areas, such as the visual cortex (Nauhaus et al., 2009, for review see Sato et al., 2012), the olfactory bulb (Freeman, 1978; Friedrich et al., 2004), or the thalamus (Kim et al., 1995).

However, the type of wave activity observed in motor cortex differs from the types of travelling waves described, for instance, in visual cortex by using optical imaging techniques (Muller et al., 2014). These authors described a solitary bump of elevated activity either induced by stimulation or propagating spontaneously from a central hotspot outwards. In contrast, motor cortical waves observed in recordings from multi-electrode arrays turn out to be rather unidirectional throughout the entire cortical territory covered by the array. These waves travelling homogeneously along a defined direction are generally called planar waves. The probability of observing these planar waves may rapidly change as a function of behaviour. Indeed, the authors of Rubino et al. 2006 found that the average coherence of phase gradients across electrodes, considered as being a signature of planar wave propagation, was highest at the beginning of the instructed delay of a center-out reaching task.

The planar waves described in Rubino et al. 2006 represent the most salient type of dynamic pattern formation, due to the unidirectional nature of its gradient field. However, potential different

spatial organization of beta oscillation patterns outside periods of planar waves has not yet been described. It is reasonable to assume that oscillatory activities do exhibit other types of patterns commonly associated with theoretical systems displaying pattern formation (e.g., Ermentrout and Kleinfeld, 2001; Heitmann et al., 2012), such as divergences or singularities. In visual cortical area MT of the marmoset monkey, for instance, the authors of Townsend et al. (2015) described a variety of patterns in slow (delta) oscillations. The power of these slow oscillations was related to the probability of observing particular types of patterns. The occurrences of motor cortical planar waves seem to be of very short duration, in the order of $50\,ms$, as noted by the authors of Rubino et al. 2006 in their Supplementary Material. This is evocative to the finding that motor cortical beta oscillations strongly modulate their amplitude by exhibiting short-lasting high amplitude packages of a few oscillatory cycles, the so-called spindles (Murthy and Fetz, 1996a,b). Even though an individual beta spindle lasts far longer than the occurrence of a planar wave, their dynamic behaviour suggest them being correlated.

Based on the findings of this previous literature, we derive three main goals for the research analysis we conduct on selected datasets of the R2G experiment: The *first goal* is to explore the possible presence of other wave-like spatio-temporal patterns than planar waves. The *second goal* is to relate patterns to behaviour in order to test their possible functional implication. The *third goal* is to test whether or not patterns are related in to modulations in beta power, i.e., the occurrence of spindles.

4.2 Material and Methods

cnd1			**cnd2**		
monkey T(t)	**monkey L**	**monkey N(i)**	**monkey T(t)**	**monkey L**	**monkey N(i)**
t010910-001	l101013-002[2,3]	i140613-001[3]	t100910-001	l101013-005[2]	
t010910-003	l101015-001[1,2,3]	i140616-001[3]	t100910-002	l101015-002[2]	
t020910-001	l101106-001	i140617-001[3]	t100910-003	l101106-003	
t020910-004	l101108-001[1,3]	i140627-001[3]	t130910-001	l101108-002	
t030910-003	l101110-003[1,3]	i140701-001[3]	t130910-002	l101110-005	
t060910-001	l101111-002[1,3]	i140702-001[3]	t140910-001	l101111-003	
t070910-001	l101126-002[1,3]	i140703-001[3,4]	t140910-002	l101126-003	
t220910-002	l101130-006	i140704-001[3]	t150910-001[2]	l101130-007	
t230910-001[2]	l101202-001[1]	i140710-001**	t150910-002[2]	l101202-002	
t230910-002	l101216-002[1]	i140718-001[3]	t160910-001[2]	l101216-004	
t240910-001[2]	l101217-001	i140721-002	t160910-002	l101217-002	
t240910-002[2]	l101220-002[1]	i140725-002[3]	t170910-001[2]	l101220-003	
t270910-001[2]	l110208-001[1]	i140801-001**	t170910-002	l110208-003	
t280910-001[2]	l110209-001[1]	i141117-001	t170910-003	l110209-002	
t280910-002[2]	l110415-002[1]	i140917-002	t200910-002	l110415-004	

Table 4.1 – Datasets used in the presented research publication. Except for the two datasets marked with **, which followed the iso task paradigm (cf. Section 2.1.2), the standard task paradigm (2-inst, cf. Section 2.1.1) was used for all datasets. Trial types alternated randomly in cnd1 and cnd2 dataset pools (gt:ra and ft:ra, cf. Table 2.2). Marked datasets were also used in [1]Riehle et al. (2013) and/or [2]Milekovic et al. (2015) and/or [3]Torre et al. (2016) or will be published in [4]Brochier et al. (prep). For monkey N(i) datasets with cnd2 of sufficient quality were not available.

For this research paper, datasets from all three monkeys of the R2G experiment were used (monkey T (t), monkey L (l), and monkey N (i), cf. Section 2.2). In detail, we aimed to select 15 datasets per monkey with condition setting 1 (grip-first, cnd2, cf. Table 2.2), and 15 datasets per monkey with condition setting 2 (force-first, cnd2, cf. Table 2.2), both with random trial type sequence (gt:ra and ft:ra, cf. Table 2.2) in the standard reach-to-grasp task (2-inst, cf. Section 2.1.1), for conducting

```
"""
Examples how the ReachGraspIO combines Neo data and odML metadata framework
"""

import os
import quantities as pq

import ReachGraspIO


# ============================================================================
# Load data
# ============================================================================
dataset_obj = ReachGraspIO(
    filename="/datasets/t010910-001",
    metadata_dir="/odML_collection/t010910-001.odml")

# Read in data
neo_block = dataset_obj.read_block(
    channels=range(1, 97), nsx_to_load=2, load_events=True)

# ============================================================================
# Add trial epochs to the data
# ============================================================================
param = {}
param['trigger'] = 'TS'
param['tpre'] = 1000 * pq.ms
param['tpost'] = 4000 * pq.ms

dataset_obj.add_trials_around(
    neo_block, name='mytrial', aligned_trigger=param['trigger'],
    tpre=param['tpre'], tpost=param['tpost'],
    aligned_offset=0 * pq.ms, performance_codes=[255])

# ============================================================================
# Extract AnalogSignal for all electrodes quantified as "good" by the
# quality assessment procedure for the IFC of the LFP data
# ============================================================================
asigs_of_good_electrodes = []
for asig in neo_block.segments[0].analogsignals:
    if asig.annotations['rejIFC'] is False:
        asigs_of_good_electrodes.append(asig)
```

Figure 4.1 – Code examples illustrating the combined data and metadata access. First, the script demonstrates for one dataset (`t010910-001`) how to load analogue signal data of all 96 electrodes of the Utah-Array from the corresponding ns2-file, in combination with the digital behavioural events stored in the corresponding nev-file and the dataset-specific metadata stored in the corresponding odML-file (cf. lines 11-21) via the `ReachGraspIO` (cf. Section 2.5.2.2). Second, the script demonstrates how to add trials as Neo `Epoch` objects to the loaded data via the `add_trials_around` function of the `ReachGraspIO` (cf. lines 22-34). This function allows the user to define a time window (via `tpre` and `tpost`) around a defined digital behavioural trial event as `trigger` (e.g., `TS`). Furthermore, the function allows the used to specify if all or only trials with a specific metadata annotation are used to create the `Epochs` (e.g., trials with `performance_code` `255`, cf. Table 2.5). Third, the script demonstrates how to generate a list of `AnalogSignal` objects from electrodes, which passed the quality assessment procedure (cf. Section 2.4.4.3 and s16 in Section 2.5.1) for the intermediate frequency components (`IFC`) of the data.

the research analysis (for lists of used datasets see Table 4.1). At the same time, to be able to more reliably relate the research findings to previous work, it was (at least) tried to match datasets used in another research publication on the data (Torre et al., 2016), which investigated the emergence of precise spike synchronization in data of the reach-to-grasp experiment using a method previously developed in the SN-group (Torre et al., 2013). This limited the selection further to the currently available offline spike sorted datasets, although the presented analysis only uses LFP signals. For monkey N these data selection criteria proved to be difficult, because (i) only 13 datasets of task type [2-inst, cnd1, gt:ra and ft:ra] were offline spike sorted, and (ii) no datasets of task type [2-inst, cnd2, gt:ra and ft:ra] of sufficient quality were available. The latter excluded monkey N from the corresponding analysis in the research paper, while the 13 datasets of task type [2-inst, cnd1, gt:ra and ft:ra] were complemented with two datasets of the very similar task type [iso, cnd1, gt:ra and ft:ra] (cf. Section 2.1.2) to even the numbers of used datasets between the monkeys. The analysis of this research paper was conducted solely on the recorded LFP data of the R2G experiment (ns2 with neuronal data, cf. Section 2.3.3), it was therefore not necessary to reduce the dataset selection further to the availability of raw data (ns5 or ns6 with neuronal data, cf. Section 2.3.3). Although the selection of datasets listed in Table 4.1 was mainly a result of a manual inspection of the corresponding metadata collection defining the wanted criteria (cf. Section 3.3.1), an alternative could have been to automatize the selection procedure by writing a script similar to the presented example codes in Section 3.3.3.

In general, data and metadata used in the analysis procedures presented in the following subsections of this chapter were accessed via the Neo data and odML metadata framework implemented for the R2G experiment (cf. Section 1.3, Section 2.5.2, and Section 1.2, Chapter 3, respectively). Simple code examples, which illustrate how data are accessed in combination with corresponding metadata in the course of the analysis via the `ReachGraspIO`, are displayed in Figure 4.1. To organize the analysis workflow for the research publication, separate scripts for analysing the data, pooling the data or analysis results across the datasets, and plotting were implemented. Part of the used analysis routines originated from the electrophysiology analysis toolkit, Elephant[1] (cf. Section 1.3). Elephant is an open-source, community centred Python library which provides for Neo data structures a consistent analysis framework as platform to share code across different laboratories (Denker et al., 2015a,b). All used and implemented code was version controlled using the open-source distributed version control system Git[2](Chacon and Straub, 2014). Parameter settings for each script were saved together with the analysis results in an hdf5 format, but in a separate files. This approach allowed us to easier infer conditional parameters from files produced by previous scripts in subsequent analysis compilations when necessary. In addition, we partially implemented unit-tests for low-level analysis functions, to control the sanity of the code in the course of the development of the complete analysis procedure.

The, for the research publication, needed descriptions of the experiment, the behavioural task, the subjects, the implant procedure and array locations, as well as the recording methods were partially based on previous publications of the R2G experiment (Riehle et al., 2013; Milekovic et al., 2015), but mostly on the data descriptor presented in Chapter 2 and are therefore not repeated here.

4.2.1 Power spectra and coherence

Power spectra and coherence were calculated using Welch's average periodogram algorithm using the `psd` and `csd` functions of the Python package `scipy`. We used windows of length $l = 1024$ sample

[1] `http://neuralensemble.org/elephant/`

[2] `https://git-scm.com/`

points (at $1\,kHz$ sampling). Each window was tapered using a Hanning window. The time-resolved power spectra (`spectrograms`) were calculated using windows of length $l = 512\,samples$ and an overlap of $500\,samples$.

4.2.2 Definition of vector fields

We calculated four different types of vector fields in order to visualize the spatial arrangement of oscillatory activity in the beta range on the array, and to provide a starting point for calculating multiple measures that characterize the arrangement. In a first step, we filtered the LFP signal on each electrode using a third-order Butterworth filter (pass band: $13-30\,Hz$) in a way that preserved the phase information (`filtfilt` function of the Python package `scipy`). The filter setting was intentionally chosen broad such that it enabled us to identify the phase and the amplitude despite temporal variations of the beta oscillation amplitude and its centre frequency. In order to compare the relative changes in amplitude between different electrodes, the amplitude of the LFP signal was then normalized across recording electrodes by computing the z-transform of the complete filtered LFP signal on an electrode-by-electrode basis.

In a next step, we calculated the instantaneous amplitude and phase of the normalized, filtered LFP time series $x_i(t)$ on each electrode i by first constructing the analytic signal $X_i(t) = x_i(t) + j\mathscr{H}(\mathrm{x_i}(t)]$, where $\mathscr{H}(\cdot)$ represents the Hilbert transform and $j^2 = -1$. From $X_i(t)$, we obtained the instantaneous signal amplitude $a_i(t)$ by taking its modulo, and the instantaneous phase $\varphi_i(t)$ by taking its argument (angle). We defined the maps $A_{xy}(t) = a_i(t)$ and $\Phi_{xy}(t) = \varphi_i(t)$ by the instantaneous phases of the LFP, where $x \in 0,...,9$ and $y \in 0,...,9$ are the coordinates of the recording electrode i of the Utah-Array.

In a further step we investigated whether, locally at each electrode and at each point in time, there is a spatially structured arrangement of the phases $\Phi_{xy}(t)$. To this end, in the remainder of this section, we defined three additional maps that we term ***phase gradient map*** $\Gamma_{xy}(t)$, ***directionality map*** $\Delta_{xy}(t)$, and ***gradient coherence map*** $\Lambda_{xy}(t)$.

The local spatial phase gradient at position electrodes (x,y) was estimated based on a neighbourhood $\bar{\aleph}_{xy}$ of its $k-$*nearest* neighbours. For border electrodes ($x \notin 2,...,7$ or $y \notin 2,...,7$), only existing electrodes were considered as nearest neighbours. In this manuscript we chose $k = 2$ to obtain a smooth map of the local phase gradients. Let $\bar{N}_{xy}$ denote the cardinality of the set $\bar{\aleph}_{xy}$. From the average gradient $d|\varphi(t)|/dx \cdot e^{j\alpha}$, between electrode (x,y) and each if its neighbours (i,j), where α denotes the angular direction between the electrodes, we now constructed the ***phase gradient map***:

$$\Gamma_{xy}(t) = \bar{N}_{xy}^{-1} \sum_{(i,j)\in\bar{\aleph}_{xy}} \frac{\Phi_{ij}(t) - \Phi_{xy}(t)}{\sqrt{(i-x)^2 + (j-y)^2}} \cdot e^{j\alpha_{ij}} \approx \nabla\Phi_{xy}(t). \tag{4.1}$$

Based on the average frequency of the beta oscillation f_β, we can easily derive the phase velocity field $\Psi_{xy}(t) = 2\pi f_\beta \Gamma_{xy}(t)$, which indicates the phase velocity of a planar wave front running through the point (x,y). Here, $f_\beta = 21.5\,Hz$ was chosen as the mean frequency of the respective beta bands of the monkeys (see Section 4.3.1 and Figure 4.4). An estimate of the macroscopic phase velocity can be obtained by averaging $v(t) = \Gamma_{xy}^{-}(t)$. Next, we defined the ***phase directionality map***

$$\Delta_{xy}(t) = |\Gamma_{xy}(t)^{-1}|\Gamma_{xy}(t) \tag{4.2}$$

by normalizing the the vectors of the phase gradient map $\Gamma_{xy}(t)$ to unit length. It indicates only the direction of the local phase gradient, independent of its magnitude. Finally, as an average of the directionality map in a neighbourhood $\aleph_{xy}$of all $k-$*nearest* neighbours of cardinality N_{xy}, we defined

the ***gradient coherence map***:

$$\Lambda_{xy}(t) = N^{-1} \sum_{(i,j)\in\aleph_{xy}} \Delta_{ij}(t). \tag{4.3}$$

It represents a second order measure of the gradient field and serves two purposes. The direction of each entry in $\Lambda_{xy}(t)$ provides a smoothed version of the vector field $\Gamma_{xy}(t)$, which is better suited for visualization due to the rather sparse sampling of activity. More importantly, the magnitude of the vectors in in $\Lambda_{xy}(t)$ indicate whether, locally, phase gradients point in the same direction (independent of the magnitudes of the gradients).

4.2.3 Quantification of observed phase patterns

Based on the phase map $\Phi_{xy}(t)$ and the three vector fields $\Gamma_{xy}(t)$, $\Delta_{xy}(t)$, and $\Lambda_{xy}(t)$ defined in Section 4.2.2, we now introduced six measures that quantitatively describe the spatial arrangement of phases on the array at each time point. These measures will later on serve as a basis to classify the phase pattern, i.e., the spatial arrangement of phases in $\Phi_{xy}(t)$, in an automatized manner. In the following, let $\hat{\aleph}$ denote the set of all used electrodes in a given recording, and $\hat{N} = |\hat{\aleph}|$ its cardinality.

Circular variance of phases One phase pattern commonly observed is one where all electrodes are fully synchronized at near-zero phase lag. Therefore, we introduced the *circular variance of phases*:

$$\sigma_p(t) = \hat{N}^{-1} \sum_{(i,j)\in\hat{\aleph}} e^{j\Phi_{ij}(t)} \in [0,1]. \tag{4.4}$$

This measure was used to determine the similarity of the phase across the array. Here, $\sigma_p(t) = 0$ indicates that an identical phase $\Phi_{xy}(t)$ observed at each electrode, whereas $\sigma_p(t) = 1$ indicates that phases are uniformly distributed across the array.

Circular variance of phase directionality In order to measure the degree to which phase gradients are globally aligned across the grid, we introduced the *circular variance of the phase directionality* (cf. top panels in A of Figure 4.2):

$$\sigma_g(t) = \hat{N}^{-1} \sum_{(i,j)\in\hat{\aleph}} e^{j\Delta_{ij}(t)} \in [0,1]. \tag{4.5}$$

A perfect planar wave is observed if $\sigma_g(t) = 0$, i.e., all phase gradients point in the same direction (independent of the magnitude of the gradients). This measure is similar to the PGD measure defined by Rubino et al. (2006).

Local gradient coherence In order to determine whether locally (within $\aleph_{xy}$) phase gradients point in a particular direction (cf. B of Figure 4.2), we considered the average length of the vectors forming the gradient coherence vector field $\Lambda_{xy}(t)$ and defined the *local gradient coherence*:

$$\mu_c(t) = |\hat{N}^{-1} \sum_{(i,j)\in\hat{\aleph}} e^{j\Lambda_{ij}(t)}| \in [0,1]. \tag{4.6}$$

$\mu_c(t) = 1$, if in each neighbourhood all phase gradients are perfectly aligned. In particular, we note that if $\sigma_g(t) = 1 \Rightarrow \mu_c(t) = 1$.

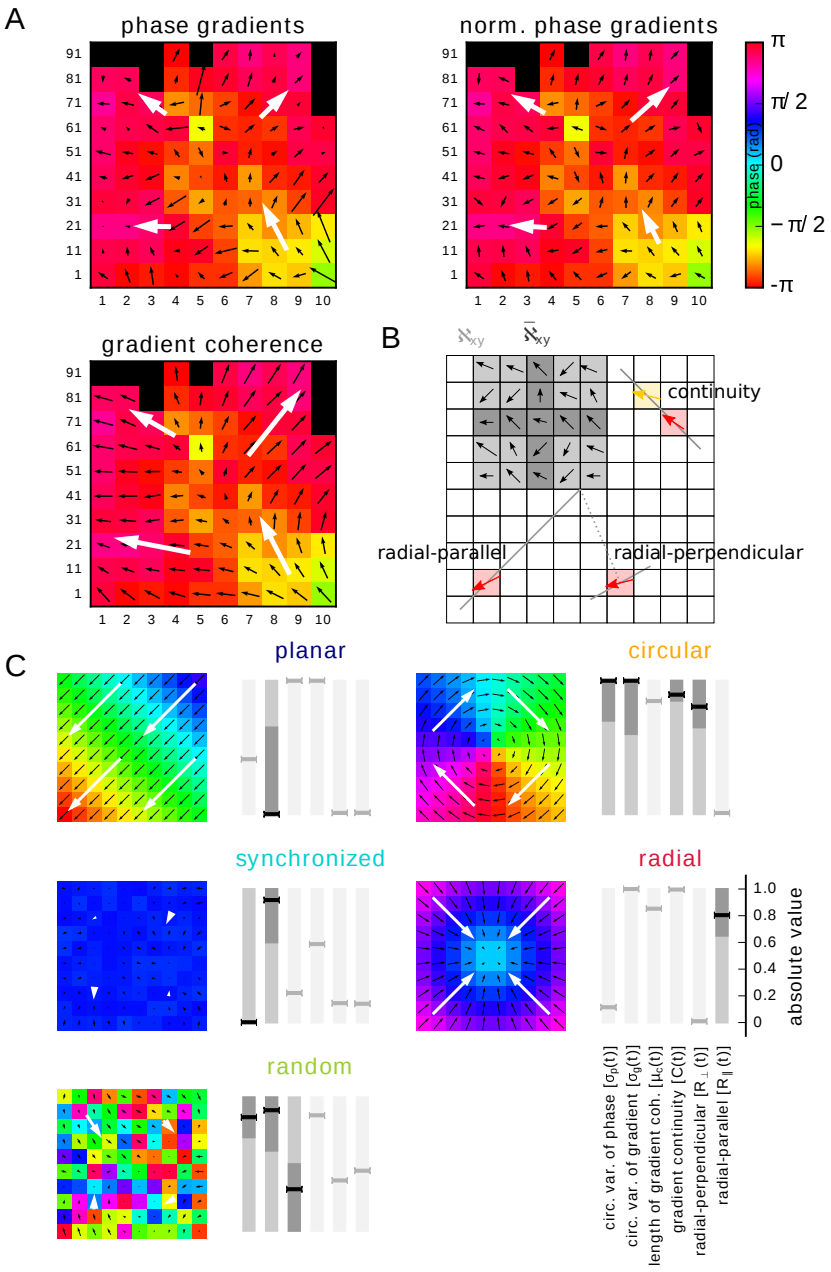

Figure 4.2 – Explanation of the methodology. (based on figure 7 of the unpublished manuscript Denker et al., prep) **A)** Example phase map (colours) of monkey L (dataset l101013-002) with overlay of phase gradient map $\Gamma_{xy}(t)$ (top left), phase direction map $\Delta_{xy}(t)$ (top right) and gradient coherence map $\Lambda_{xy}(t)$ (bottom left) shown as black arrows per electrode. White large arrows: corresponding quadrant-averaged maps. **B)** Sketch of the $10x10$ electrode array (squares) to illustrate the neighbourhoods $\aleph_{xy}$ (light grey) and $\bar{\aleph}_{xy}$ (dark grey), and the calculation of the gradient continuity $C(t)$ (top right), radial-parallel alignment $R_{\parallel}$ (bottom left), and radial-orthogonal alignment $R_{\perp}$ (bottom right). $C(t)$: angle between the gradient (red arrow) at a chosen electrode (red square) and the gradient at the square that the red gradient points to (yellow arrow and square). $R_{\parallel}$ and $R_{\perp}$: angle between the gradient (red arrow) at a chosen electrode (red square) and the corresponding solid grey line (parallel to and perpendicular to a line through the array centre and the electrode, respectively). **C)** Calculated values for the six measures (represented by horizontal line markers; from left to right: $\sigma_p(t)$, $\sigma_g(t)$, $\mu_c(t)$, $C(t)$, $R_{\parallel}$, and $R_{\perp}$) for artificial data modelling ideal realizations of the five phase patterns (*planar wave*, *synchronized*, *random*, *circular*, and *radial patterns*). A small amount of noise was added to the synchronized phase pattern to avoid division by zero in calculating measures. Darker backgrounds and black markers: measure was used to detect the corresponding phase pattern. Darkest areas indicate the range of valid values for each measure required for a positive classification according to the threshold criteria (cf., Table 4.2). Light backgrounds and grey markers: measures that are not used to detect the corresponding pattern. Region of valid values for measure $\sigma_p(t)$ for synchronized patterns is close to zero.

Gradient continuity Along the same argument we may ask even more strictly whether phase gradients locally not only point in a similar direction, but whether in fact they tend to form continuous lines. To this extend we defined in the following the *gradient continuity* $C(t)$ (cf. B of Figure 4.2). For each point (x,y), we determined the direction of the phase gradient from $\Delta_{xy}(t)$ and calculated the electrode at position (i,j) that the gradient points to. We then calculated the scalar product $s_{xy} = \left\langle \Delta_{xy}(t), \Delta_{ij}(t) \right\rangle$, which equals 1 if the two lines form a continuous line, and equals 0 if the gradients are orthogonal to each other. When averaging across all electrodes we obtain the *gradient continuity*:

$$C(t) = \hat{N}^{-1} \sum_{(i,j)\in\hat{\aleph}} s_{xy} \in [-1,1]. \tag{4.7}$$

Circular arrangements Besides spatial phase patterns that appear either homogeneous or linear, we observed patterns that appear radial or circular with respect to the centre of the array. To capture such spatial arrangements, we defined the two measures radial-parallel alignment $R_{\parallel}$ and radial-orthogonal alignment $R_{\perp}$ (cf. B of Figure 4.2). We first constructed a vector l_{xy} normalized to length $|l_{xy}| = 1$ pointing from the centre of the array, at position $(4.5, 4.5)$, to each electrode at position (x,y). We then defined the scalar product $s^{\parallel}_{xy}$ between the phase gradient $\Delta_{xy}(t)$ at (x,y) and l_{xy} , and the scalar product $s^{\perp}_{xy}$ between $\Delta_{xy}(t)$ and the perpendicular direction $l_{xy}e^{j\pi/2}$. With this, the following two equations result:

$$R_{\parallel} = \hat{N}^{-1} \sum_{(i,j)\in\bar{\aleph}} |s^{\parallel}_{xy}| \in [0,1], \tag{4.8}$$

$$R_{\perp} = \hat{N}^{-1} \sum_{(i,j)\in\bar{\aleph}} |s^{\perp}_{xy}| \in [0,1]. \tag{4.9}$$

For these equations, $R_{\parallel} = 1$ indicates that phase gradients point inward or outward, as observed for a radial phase pattern, whereas $R_{\perp} = 1$ indicates that phase gradients point in the orthogonal direction, as observed for a circular phase pattern (cf. B and C of Figure 4.2).

4.2.4 Classification of spatial phase patterns

Equipped with the six measures defined in Section 4.2.3, we were now able to classify the spatial pattern observed at each point in time according to the salient patterns that were visually observed. The patterns we distinguished are termed the ***planar wave***, ***synchronized***, ***random***, ***circular***, and ***radial*** patterns. A detailed description of these patterns are presented in Section 4.3.2.

Threshold-based classification A straight-forward method to disambiguate the five phase patterns (*planar wave*, *synchronized*, *random*, *circular*, and *radial patterns*) is to define for each pattern specific thresholds $\theta_1 - \theta_8$ on those measures that capture relevant characteristics of the pattern. Specifically, θ_1 and θ_2 are thresholds on $\sigma_p(t)$, θ_3 and θ_4 on $\sigma_g(t)$, θ_5 on $\mu_c(t)$, θ_6 on $C(t)$, θ_7 on $R_{\perp}$, and θ_8 on $R_{\parallel}$ (cf. C of Figure 4.2). The threshold conditions for each phase pattern are summarized in Table 4.2. In the following we summarize the rationale for choosing these threshold conditions.

Planar wave patterns are characterized by a non-zero phase-gradient that points in the same direction at each electrode, and are thus well characterized by a small value of $\sigma_g(t)$.

Perfectly ***synchronized*** patterns exhibit the same phase at each electrode (small $\sigma_p(t)$), and the grid-averaged phase gradient is random (large $\sigma_p(t)$).

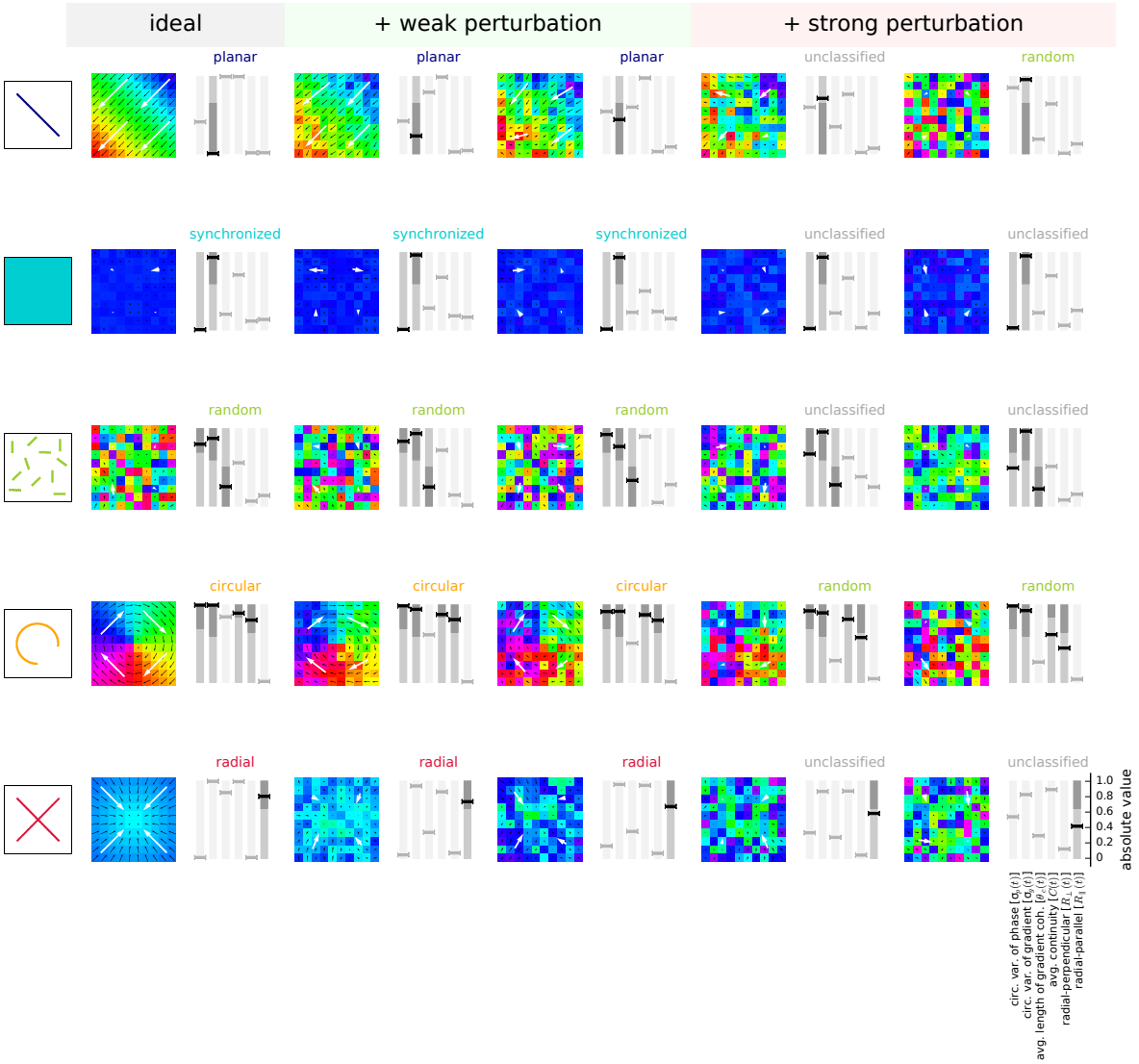

Figure 4.3 – Selection of thresholds used for automatic phase pattern classification. (based on figure S-2 of the unpublished manuscript Denker et al., prep) For each of the idealized planar wave, synchronized, random, circular, and radial patterns (rows) five artificially generated realizations are shown (cf. Figure 4.2): one ideal pattern (left column), two patterns with weak perturbation (weak additive noise) which we considered as valid representatives of that pattern type (second and third column), and two patterns with strong perturbation (strong additive noise) which we did not consider as valid representatives of that pattern type (fourth and fifth column). Absolute values of the six measures are shown to the right of each pattern (represented by horizontal bar markers; from left to right: $\sigma_p(t)$, $\sigma_g(t)$, $\mu_c(t)$, $C(t)$, $R_\parallel$, and $R_\perp$). Darker backgrounds and black markers: Measure was used for classification for the corresponding pattern. Darkest areas mark valid values of the measure that lead to a successful classification of the pattern according to the threshold criteria (cf. Table 4.2). Light backgrounds and grey markers: Measure is not used for classification of the pattern treated in the corresponding row. Text above panels: result of automatic pattern classification. Due to the specific choice of destroying patterns by adding noise (as opposed to, e.g., randomizing phase gradients), not every measure is necessarily affected by increased noise levels. Exceptions: (i) Ideal synchronized pattern (row 2, column 1) has minimal additional noise to avoid division by zero when calculating measures, (ii) for random patterns (row 3), the amount of additive noise is decreased from left to right to drive the state from fully randomized phases towards the synchronized state.

phase pattern	threshold conditions			
planar wave	$\sigma_g(t) < \theta_3$			
synchronized	$\sigma_p(t) < \theta_1$	$\sigma_g(t) \geq \theta_4$		
random	$\sigma_p(t) \geq \theta_2$	$\sigma_g(t) \geq \theta_4$	$\mu_c(t) \leq \theta_5$	
circular	$\sigma_p(t) \geq \theta_2$	$\sigma_g(t) \geq \theta_4$	$C(t) \geq \theta_6$	$\lvert R_\perp \rvert \geq \theta_7$
radial	$\lvert R_\parallel \rvert > \theta_8$			

Table 4.2 – Overview of threshold conditions used for phase patterns. (based on table 2 of the unpublished manuscript Denker et al., prep) A time point was classified as one of the 5 listed phase patterns if all corresponding threshold criteria were met. Tests were administered in the order given by the table, and the first match is chosen as the classification.

Random patterns, on the other hand, show a random phase at each electrode (large $\sigma_p(t)$), and also the phase gradient is random not only when considering the complete grid (large $\sigma_g(t)$), but also locally in each neighbourhood $\aleph_{xy}$ of an electrode (large $\mu_c(t)$).

Circular patterns share the feature of displaying all phases and phase gradients across the array with the random pattern (large $\sigma_p(t)$ and $\sigma_g(t)$), however, they exhibit a high continuity of the gradient coherence field (high $C(t)$) and phase gradients are arranged in a circular fashion around the grid centre (high $R_\perp$).

Similarly, ***radial*** patterns exhibit phase gradients that are arranged in an outward or inward pointing direction with respect to the grid centre (high $R_\parallel$), however, in contrast to the circular pattern the distributions of phases and phase gradients, and the gradient continuity are less characteristic of this pattern and, therefore, are not used.

Based on these conditions, the thresholds for the five phase patterns were set empirically in such a way that they led to a conservative association of each pattern with its corresponding pattern class, i.e., only clearly identified patterns were classified (cf. Figure 4.3). In this study the following values were used (cf. C of Figure 4.2): $\theta_1 = 0.15$, $\theta_2 = 0.7$, $\theta_3 = 0.5$, $\theta_4 = 0.6$, $\theta_5 = 0.5$, $\theta_6 = 0.85$, $\theta_7 = 0.65$, and $\theta_8 = 0.65$.

4.3 Results

The results, we present in the following, are retrieved by performing the previous described analyses on the datasets listed in Table 4.1. The task and time line of the behavioural trials performed in these datasets is summarized in Section 2.1.1 and Section 2.3.2, and displayed in Figure 2.1 and Figure 2.6. On average the recording lasted about $15\,min$ for each dataset and contained 120–140 correct trials from a random alternating trial type sequence. As a reminder, while the monkeys performed the task, massively parallel neuronal activities were recorded using a 100-electrode Utah-Array (Blackrock Microsystems, Salt Lake City, UT, USA) implanted in the contra-lateral primary motor (MI) and premotor (PM) cortices with respect to the active arm (monkey L and N(i), left hemisphere, and monkey T(t), right hemisphere). The detailed locations of the implanted arrays are shown in Figure 2.3 (for monkey L and N(i)) Figure 2.2 (for monkey T(t)).

In this study we concentrate on the local field potential signals (LFPs), filtered between $0.3-250\,Hz$ and sampled at $1\,kHz$. As previously elaborated (cf. introduction of Section 4.2), we selected for further analysis in each monkey 15 datasets from the grip-first condition 1, and additionally 15 datasets from the force-first condition 2 in monkey L and T, respectively (cf. Table 4.2). In the following, we will start by characterizing the spectral properties of the recorded LFP activity to identify its oscillatory features, before characterizing these oscillations also in the spatial domain.

4.3.1 Spectral LFP properties

On a first glance, we observed that the LFP in all monkeys was dominated by a prominent oscillatory component in the beta range (about $15-35\,Hz$) (see B of Figure 4.4). By computing the average power spectrum of each monkey's LFP, pooled for one electrode across its complete set of recordings in the grip-first condition (15 per monkey), we found that the frequency range of the beta oscillation varied between monkeys (see A of Figure 4.4). Based on these spectra we defined a wide frequency band ($13-30\,Hz$) that was common to all monkeys and covered the peaks of the individual beta frequencies (shaded area in A of Figure 4.4). For allow for a better comparison, in this study we applied this same filter band in the beta range to all data sets of all monkeys.

Furthermore, the observed LFP activity revealed that the power of the oscillatory activity was not stationary in time, but was strongly modulated during the trial. Using the same electrodes for each monkey as for the averaged power spectra (cf. A of Figure 4.4), the strength of the beta oscillations is visualized by the time-resolved power spectra averaged across all successful SG trials of one representative recording session of monkeys L, T and N, respectively (cf. B of Figure 4.4). The beta power revealed a characteristic temporal evolution (for review see Kilavik et al., 2013) that was comparable across monkeys: the beta power was largest around the cue, and decayed gradually during the delay period and was strongly attenuated during movement execution. During movement, a low frequency signal was the most prominent component in the LFP, corresponding to the movement-related potential (see Riehle et al., 2013).

The inspection of single-trial LFP signals revealed that, in addition to the beta power modulations observed in trial averages, single trial LFP signals exhibit a modulation of the amplitude of beta activity (cf. A of Figure 4.5) on a much shorter time scale. Such epochs of increased beta activity comprising about 5-7 cycles (i.e., about $300\,ms$) are commonly referred to as beta spindles. During a single trial, LFP signals recorded in parallel from all electrodes of the Utah-Array did not reveal obvious spatial differences (cf. B of Figure 4.5), and in particular spindles occurred simultaneously on all electrodes (cf. also Murthy and Fetz, 1996a). However, across trials spindles did not reoccur at the same points in time (cf. A of Figure 4.5), but instead their occurrence in time exhibited a strong degree of variability. Therefore, the trial-averaged temporal evolution of beta power (cf. B of Figure 4.4) most likely represents a measure that confounds the probability of spindle occurrence, their average duration, and their average maximal amplitude.

4.3.2 Identification of phase patterns

Having described the principle properties of beta oscillations, which represent the dominant oscillatory mode of LFP activity on a single electrode and therefore offer the most tractable description of the population activity, we are now in a position to investigate the fine spatial patterning of this activity across all electrodes of the array.

When zooming in time into the LFP signals recorded from a few neighbouring electrodes during the entire trial length (cf. B of Figure 4.5) and showing the same, but beta-filtered (red traces) signals during a short selected time window (cf. C of Figure 4.5), we observed that despite a high degree of similarity, the oscillatory components express a small time-lag between each other (compare blue markers on each trace indicating oscillation peaks and troughs). To understand if there is a specific spatial patterning of the temporal lags between the signals on the different electrodes, we decomposed the beta-filtered LFP time series of each electrode i into the instantaneous amplitude $a_i(t)$, corresponding to the envelope of the filtered signal, and phase $\varphi_i(t)$ of the beta oscillation by calculating its analytic signal (see Section 4.2.2). We then displayed these quantities as spatial

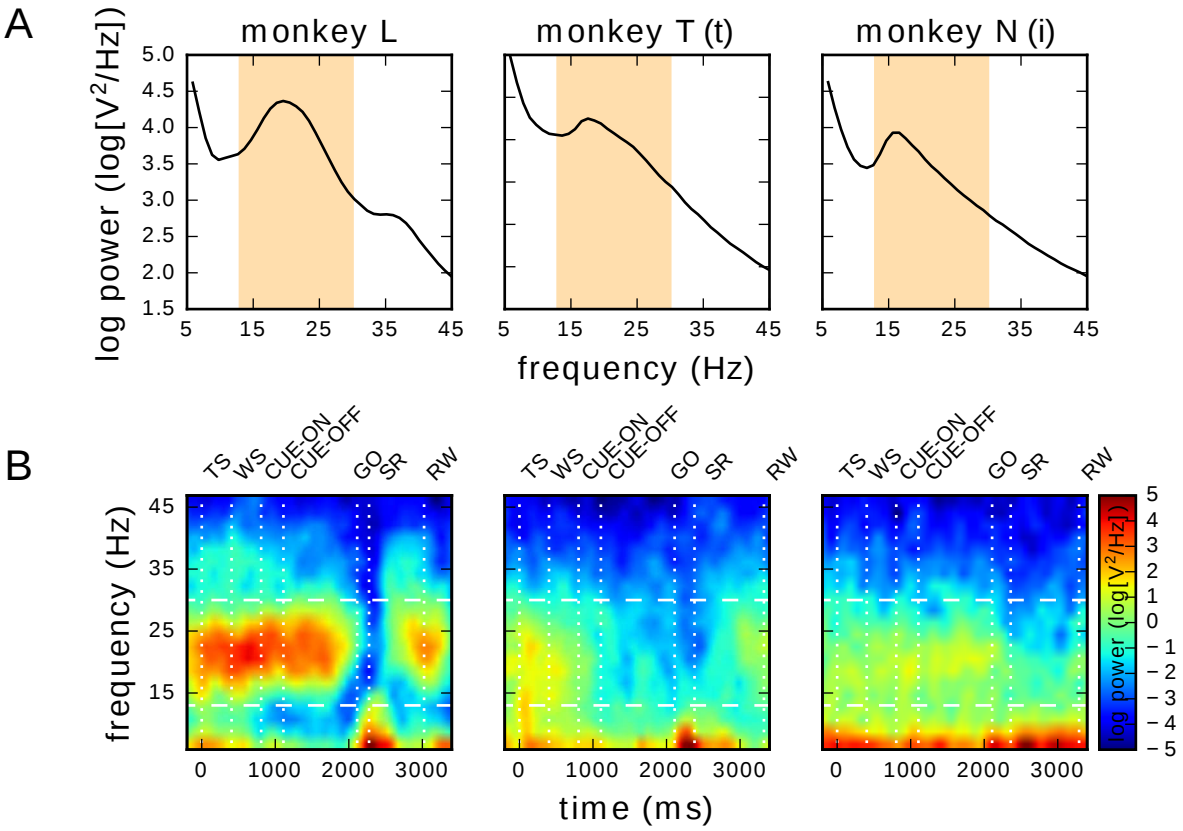

Figure 4.4 – **Spectral properties of the LFP.** (based on figure 1 of the unpublished manuscript Denker et al., prep) **A)** Power spectrum of the LFP during the complete recording of one selected electrode averaged across all datasets (N=15 per panel) in the grip-first condition 1 of monkey L (left), T(t) (middle), and N(i) (right). Orange shading: range of the applied beta-band filter (the ends mark the cut-off frequencies). **B)** Trial-averaged, time-resolved power spectrogram of the LFP of one electrode in one dataset during an SG trial of a grip-first recording. Trials aligned to TS. Colour indicates logarithmic power density. Horizontal dashed lines: beta band as shown in panel A. Vertical dotted lines: trial events with mean times for SR and RW (for event abbreviations cf. Section 2.1.1 and Section 2.3.2). Used datasets: l101013-002 for monkey L, t010910-001 for monkey T(t), and i140613-001 for monkey N(i), from left to right, respectively.

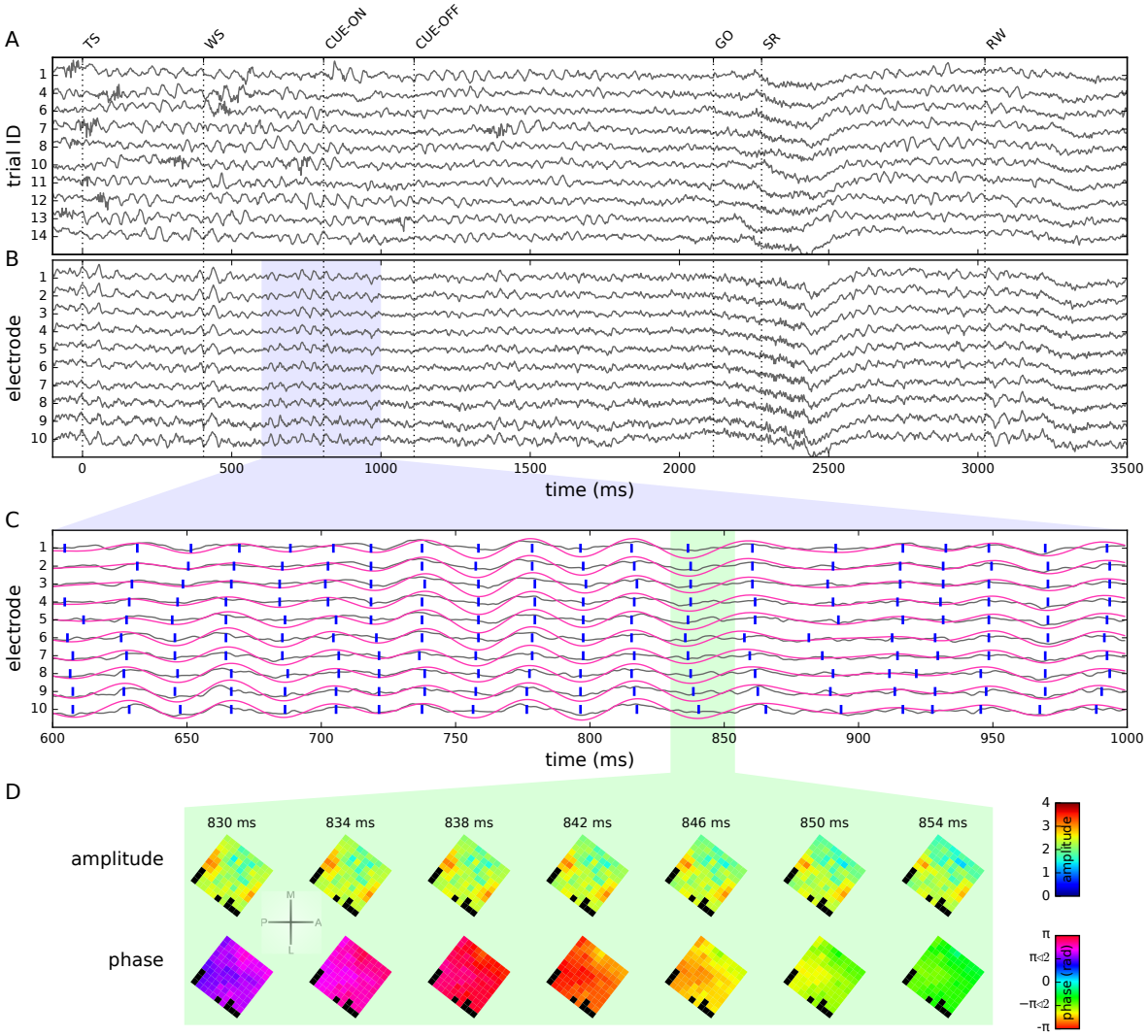

Figure 4.5 – Extraction of phase and amplitude maps. (based on figure 2 of the unpublished manuscript Denker et al., prep) **A)** LFPs (z-scored) recorded from one single electrode during 10 consecutive successful trials (monkey L, dataset: l101013-002). Trials aligned to $TS = 0\,ms$. **B)** Simultaneously recorded LFPs from 10 neighbouring electrodes on the Utah-Array during a single trial. **C)** Blow-up of the LFPs of the 10 example electrodes shown in panel B (grey traces; blue shading in panel B indicates selected time window). Red traces: beta-filtered LFP. Blue lines: locations of peaks and troughs in the filtered LFP, i.e., phases $\varphi = 0$ and $\varphi = \pi$. **D)** Amplitude (top) and phase (bottom) maps (shown in $4\,ms$ steps) recorded during a $24\,ms$ window (green shading in panel C). Colour in each square indicates the amplitude and phase of the LFP at the electrode of a given position. Black squares: unconnected electrodes or electrodes rejected due to signal quality. The images are rotated to match the cortical position of the array as indicated in Figure 2.3.

maps $A_{xy}(t)$ and $\Phi_{xy}(t)$ for amplitude and phase, respectively, with each electrode at its spatial array position (x,y) and at each time point t. Although the oscillation amplitude $A_{xy}(t)$ was not the same across the array, it was highly correlated between electrodes and showed a pattern that was changing slowly as compared to the time scale of the beta period (cf. D of Figure 4.5). This finding matches our observation that the occurrence of spindles is coherent across recording electrodes (cf. B of Figure 4.5).

In contrast, the phase snapshots $\Phi_{xy}(t)$ showed a pronounced structure that varied on a very fast time scale in the range of milliseconds (cf. D of Figure 4.5). Upon visual inspection of the obtained plots, it was evident that the spatial patterns exhibited by the phase time series were highly dynamic and changed in time. While we typically observed a smooth transition of maps between consecutive time points t_i and t_{i+1} (given a sampling rate of $1\,kHz$), at some moments in time the maps changed their structure very rapidly. Despite the rapid changes of the spatial structure of the maps and some discontinuities in their temporal evolution, many phase maps could be classified by visual inspection into one of five distinct classes of spatial arrangements (*planar wave*, *synchronized*, *random*, *circular*, and *radial patterns*), in the following referred to as phase patterns (cf. Section 4.2.4). Representative examples of these classes of phase patterns and their temporal evolution over a time period of $20\,ms$ are shown in A and B of Figure 4.6.

In order to better visualize and characterize these spatial structures, we calculated the vector field of phase gradients $\Gamma_{ij}(t)$ and its spatially smoothed version, the phase gradient coherence $\Lambda_{ij}(t)$, and display the gradient fields along with the phase maps. In the following we will briefly now describe these classes of phase patterns in their most salient, idealized form.

The identification of planar travelling waves (cf. top row of B in Figure 4.6), comparable to the first report by Rubino et al. (2006), were most obvious. In these planar patterns, a planar wave front travelled across the array, where the spatial period was typically larger than the array dimensions. Second, we observed a synchronized pattern (cf. 2nd row of B in Figure 4.6), in which the signals on all electrodes were synchronized at near-zero phase lag. Complementing this state at the other extreme, we observed a random pattern (cf. 3rd row of B Figure 4.6), which showed no apparent phase relation between electrodes. A fourth pattern, termed circular pattern (cf. 4th row of B Figure 4.6), was characterized by an area near the array centre around which the activity revolved. Finally, we observed a radial pattern (cf. bottom row of B in Figure 4.6) of radially inward or outward propagating waves, which was also characterized by a point of origin near the array centre. A specific type of pattern persisted for only short time periods on the order of the duration of a single beta oscillation. In addition, not every phase map at any time point could be clearly attributed to one of these phase five patterns.

Given this empirical identification of classes of phase patterns, we aimed to automatically classify the sequences of phase maps into one of these classes where possible. To this end, we introduced a set of six measures (cf. C in Figure 4.6) that capture features of the spatial arrangement of beta oscillations based on the phase map $\Phi_{xy}(t)$, and its spatial arrangement quantified by the phase gradients $\Gamma_{xy}(t)$ and the gradient coherence $\Lambda_{xy}(t)$ independently at each time point t. The details of how to construct these measures are given in Section 4.2.3. Essentially, each of the measures represents a feature of a given phase pattern that is characteristic for one or several of the five classes of phase patterns. The planar patterns, described by a planar wave front travelling across the entire array, were characterized by local phase gradients that were aligned in parallel across the array. Thus, such a pattern was composed of a maximum of one wave front. The synchronized pattern was distinguished by a single phase value at all electrodes and a uniform circular distribution of phase gradients across the array. The random pattern showed no apparent phase relation between electrodes. In the circular pattern, like in the synchronized state, the phase gradients were uniformly distributed, but in contrast the

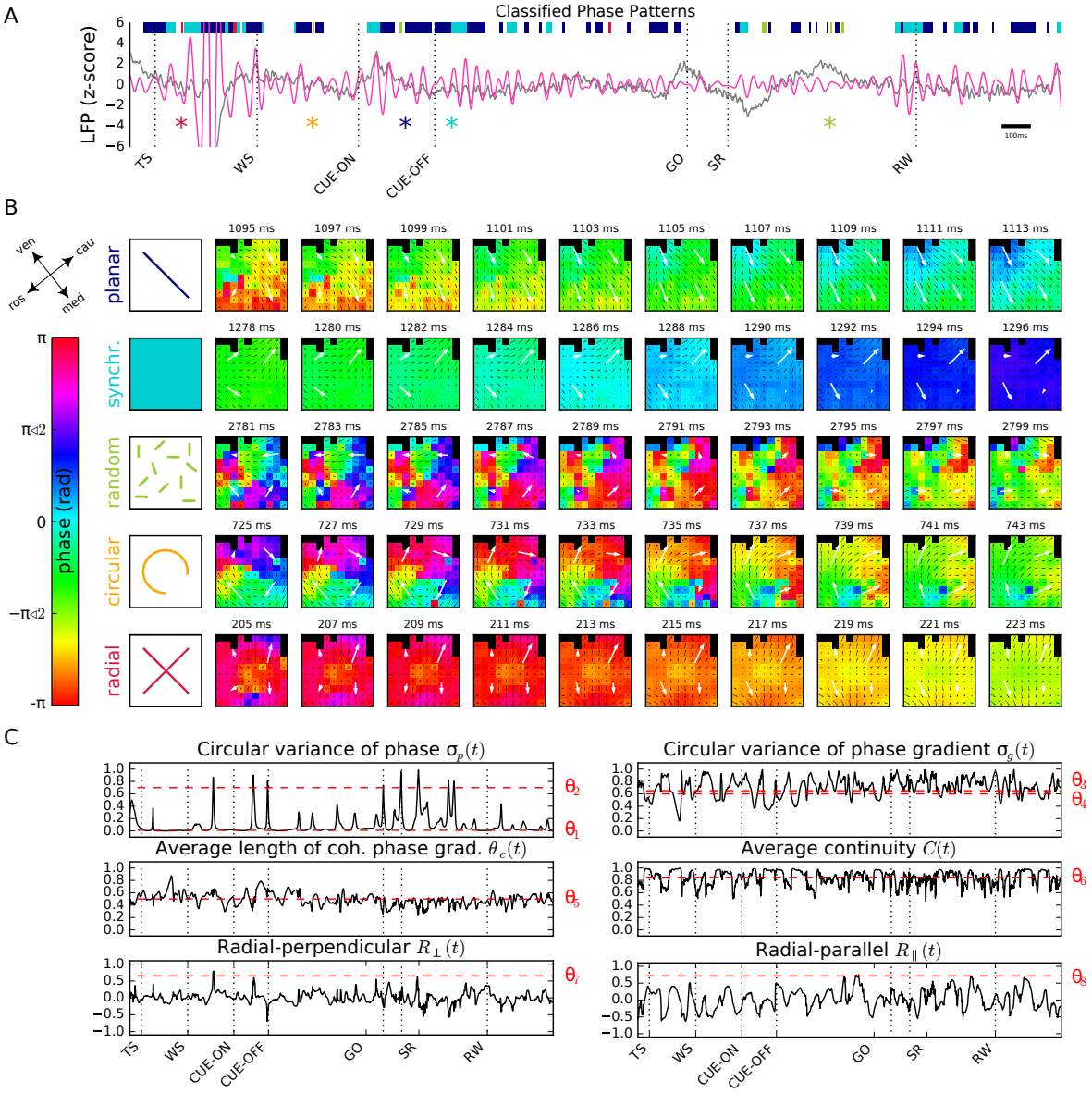

Figure 4.6 – Phase patterns and their detection. (based on figure 3 of the unpublished manuscript Denker et al., prep) **A)** LFP signal (z-scored) recorded during a single trial (monkey L, dataset: l101108-001, trial ID: 19) on a single electrode (grey) and superimposed beta-filtered LFP (red). The signal is aligned on TS. Dashed vertical lines indicate trial events. Coloured horizontal bars show time periods during which a particular type of phase pattern (compare colour code in panel B, first column) was detected. The coloured asterisks mark the time points of the first frame of the wave patterns shown in panel B. **B)** Phase maps of one example of an automatically detected phase pattern for each type of phase pattern (rows, from top to bottom: planar wave, synchronized, random, circular, and radial). The sequence of maps in one row show a total of $18\,ms$ in steps of $2\,ms$. The pattern was initially detected in the first phase map of each row (corresponding time point indicated by an asterisk in panel A). Flow field indicated by black lines: gradient coherence map $\Lambda_{xy}(t)$; white large arrows: corresponding quadrant-averaged gradient coherence shown for visualization. **C)** Graphs of the time-series (corresponding to the trial shown in panel A) of the six measures (top to bottom:$\sigma_p(t)$, $\sigma_g(t)$, $\mu_c(t)$, $C(t)$, $R_\parallel$, and $R_\perp$) used for detection of the different patterns. Red dashed lines indicate the thresholds $\theta_1 - \theta_8$ used to detect and classify the phase patterns at each time-point.

distribution of phases across the array was also uniform. Thus, phase gradients were aligned in local neighbourhoods arranged around the centre of the array. And finally, the radial pattern pointed on a global level in an orthogonal direction to lines passing the central point. Thus, phase gradients were also aligned in local neighbourhoods, as did circular patterns, but pointed in an orthogonal direction (inward or outward).

Based on the six measures, we used thresholds (cf. red dashed lines in C of Figure 4.6) to assign each phase pattern at time point t to one of the five classes of patterns, or, if none of the threshold criteria were met, the phase pattern was left unclassified. As described in Section 4.2.4, thresholds were set empirically in such a way that they led to a conservative association of phase patterns with pattern classes, i.e., only clearly identified patterns were classified (see Figure 4.2 and Figure 4.3 for a visualization of accepted and rejected classifications).

Details of the classification process are provided in Section 4.2.4. Our classification procedure may have had some technical limitations such as the low spatial sampling of the 100 electrodes ($400\,\mu m$ inter-electrode distance) and the small spatial window of observation ($4\,mm\,x\,4\,mm$) as compared to the spatial wavelengths of exhibited by some patterns. This may affect, in particular, the radial and circular patterns in which the point of origin was not at the array centre, making it impossible to infer the pattern unequivocally. Additionally, observed patterns could also have represented transient dynamics from switching between patterns or even overlaps of competing patterns, which could not be properly distinguished. If for any of these reasons a pattern did not fulfil the strict criteria of one of the five pattern classes described above, we referred to it as *unclassified*.

The use of our algorithm allowed us to specify more precisely the five phase patterns already defined by visual inspection of the phase maps. The phase patterns shown in B of Figure 4.6 were determined by using this algorithm. In Figure 4.6, A shows the LFP recorded on one electrode during one single trial, in which all classified phase patterns are marked, including those shown in panel B of the same figure. The corresponding measures and thresholds used in the classification procedure are depicted in C of Figure 4.6.

The percentage of time points identified as each of the pattern classes is given in Table 4.3, for each monkey separately, and the number of epochs of contiguous time points classified as the same pattern is displayed in A of Figure 4.7. These results show that all pattern types were observed in abundance in each monkey, with planar wave patterns being among the most prominent and circular patterns among the least observed patterns. Only in monkey N(i), the random pattern was observed more often than the planar wave pattern. In addition, monkey N(i) rarely exhibited a synchronized pattern as compared to monkeys L and T(t).

phase pattern	**monkey L**	**monkey T(t)**	**monkey N(i)**
planar wave	28.70%	42.41%	5.61%
synchronized wave	1.62%	14.98%	0.02%
random wave	1.28%	0.50%	8.89%
circular wave	0.01%	0.01%	0.02%
radial wave	1.22%	0.46%	1.41%
unclassified	67.17%	41.65%	84.05%

Table 4.3 – Occurrence of pattern classes. Percentage of time points classified as a specific phase patterns in each monkey given the conservative choice of thresholds used in the analysis.

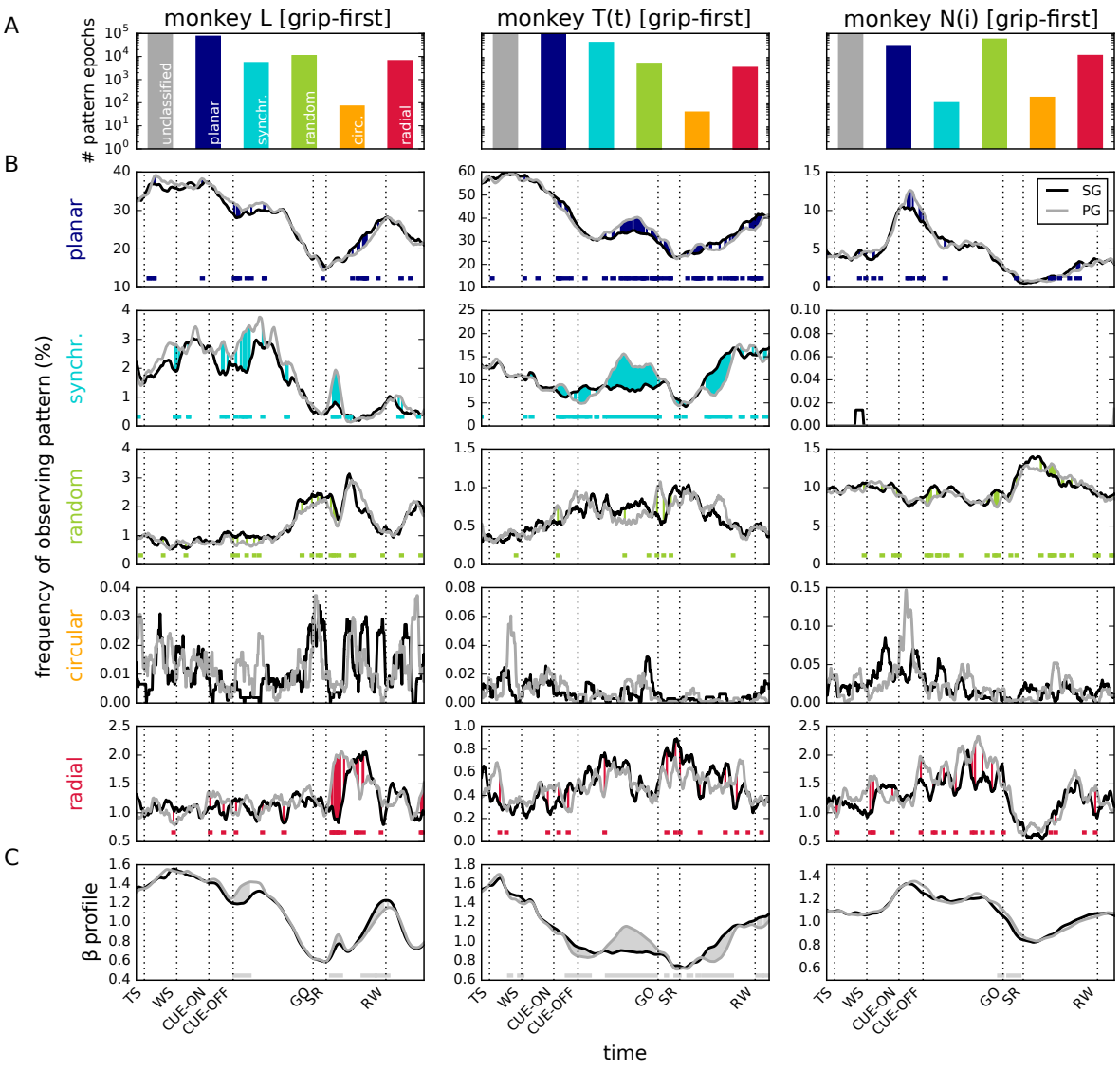

Figure 4.7 – Behavioural correlates and relation to average beta power. (based on figure 4 of the unpublished manuscript Denker et al., prep) **A)** Number of epochs of a phase pattern detected continuously for at least five consecutive time frames, i.e., $5\,ms$ (bars from left to right: planar wave, synchronized, random, circular, radial pattern; colour code: see panel B) for monkey L (left), T(t) (middle), and N(i) (right). Data were obtained from all selected datasets including inter-trial periods. **B)** Time-resolved probability of observing a specific phase pattern (rows, from top to bottom: planar wave, synchronized, random, circular, radial pattern) during the trial. Statistics were averaged across all trials of all recording sessions for each monkey (N=15) and smoothed with a box-car kernel of length $l = 100\,ms$. Only grip-first trials were selected and separated into side-grip (SG) trials (black) and precision-grip (PG) trials (grey). For monkey N(i), only very few synchronized patterns were detected during the trial. Colour shading between curves and coloured bars indicate time periods where SG and PG curves different significantly (Fisher's exact test under the null hypothesis that the number of trials where the given state was or was not observed at a given time point is independent of whether the trial was SG and PG, $p = 0.1$). **C)** Beta power profile pooled across SG (black) and PG (grey) trials (same data as in panel B). The power profile $a(t)$ of a single trial is calculated as the time-resolved instantaneous amplitude $A_{ij}(t)$ of the beta-filtered LFP averaged across all electrodes (i, j), and measures the instantaneous power of the beta oscillation in that trial. Grey shading between curves and horizontal bars indicate time periods where SG and PG curves different significantly ($t-test$ under 10 the null hypothesis that the distributions of single trial amplitudes $A_{ij}(t)$ at each time point t are identical for SG and PG trials, respectively, $p = 0.1$).

4.3.3 Relation of power and phase patterns to behaviour

In this section we investigated the precise relationship between phase patterns and behaviour. Periods of clearly identified phase patterns were typically of short duration and occurred interspersed throughout the trial. In a first step of our analysis, we counted for each monkey the number of occurrences of continuous periods each of the five phase patterns and unclassified patterns during the entire length of all selected sessions, including the behavioural trials and the inter-trial intervals. In Figure 4.7, panel A shows, for the grip-first condition, that in all monkeys planar waves were among the most frequently observed patterns. The largest amount of periods of synchronized patterns was observed in monkey T(t), and the lowest in monkey N(i), whereas the largest amount of periods of random patterns was observed in monkey N(i), compared to the two other monkeys.

We then investigated whether or not the occurrence of a specific phase pattern is linked to one or more behavioural events. In order to understand the relation between a specific pattern and behaviour, we determined trial by trial and for each pattern separately its precise occurrence during the time course of the behavioural trial. We pooled the data from each monkey across all trials of the same condition (correct trials only, cf. Section 2.3.2), i.e. grip-first condition 1 or force-first condition 2, and across all selected recording datasets, to obtain a measure for the probability of the occurrence of a specific pattern in time. In Figure 4.7, B shows that monkey N(i) had comparatively low numbers of planar and synchronized patterns during the trial, but a higher number of random patterns than the two other monkeys. This suggests that the high amount of planar and synchronized patterns of monkey N(i) presented in A of Figure 4.7 occurred during the inter-trial intervals, and not during the trial as shown in B of Figure 4.7.

In the next step, we assessed similarities in the temporal profile of the pattern occurrence probabilities (cf. B in Figure 4.7). For each monkey, the probability of observing any pattern was strongly modulated over the time course of the trial. Common to all monkeys was the finding that planar patterns occurred mostly during the initial cue presentation and during reward, and were less prominent during movement. Synchronized patterns expressed a similar time course for monkey L and to a much lesser degree for monkey T(t). Monkey N(i) showed almost no synchronized pattern during the trial. In contrast, in all monkeys random patterns occurred predominantly towards the end of the delay period or during movement. Circular and radial patterns were rarely observed during the trial. They exhibited a clear modulation structure in time for each monkey, but in a different way for each monkey.

The specific and consistent temporal modulation of the occurrence probability suggests that the spatio-temporal structure of activity is related to motor cortical processing performed during the trial. We thus asked, if also particularities of the trial condition were reflected in the probability. To this end, we compared results obtained during SG and PG conditions (cf. black and grey, respectively, in B of Figure 4.7). In general, the modulations of probability for both trial types were very similar, but expressed a few notable exceptions. For planar waves, SG and PG deviated slightly, but significantly, during early delay (probability of observing a pattern during PG trials exceeded that of SG trials, $PG > SG$) and before reward ($SG > PG$) for monkey L, during late delay ($PG > SG$) in monkey T(t), and during cue presentation for monkey N(i) ($PG > SG$). More strikingly, synchronized patterns deviated significantly during early delay ($PG > SG$) and after movement onset ($PG > SG$) for monkey L, and during late delay ($PG > SG$) and before reward ($SG > PG$) for monkey T(t). In addition, for monkey T(t) we observed the reversed effect for random patterns during late delay ($SG > PG$).

Up to now, we concentrated on analysing the time-resolved average spatial organization of oscillatory activity on the basis of the phase information extracted from the time series. We therefore asked

how these findings relate to the trial-averaged beta power. Indeed, we noticed that the temporal evolution of the occurrence probability of planar and synchronized phase patterns was reminiscent of the evolution of power in the beta range shown in the spectrograms (cf. B of Figure 4.4). To further investigate this observation, we calculated the trial-averaged beta power profiles $a(t)$, that is, the time-resolved amplitude, or envelope, $A_{ij}(t)$ of the beta signal pooled across all electrodes (i, j), as a representative of the average instantaneous strength of the beta oscillation. Again, data were calculated for all sessions used in B of Figure 4.7 and separately for SG and PG trials (cf. C of Figure 4.7). Interestingly, for all three monkeys the time-resolved beta profiles closely followed the occurrence probability of the planar phase pattern (cf. top in B of Figure 4.7). For monkeys L and T(t), also the time course of synchronized patterns loosely followed that of the beta profiles. In particular, we noticed that all differences between SG and PG trials identified in the pattern occurrence probabilities were mirrored in the beta profile: in monkey L, the beta profiles obtained in SG and PG trials differed during the early delay ($PG > SG$), mirrored in the occurrence of planar waves, and after movement onset ($PG > SG$), in the occurrence probability of synchronized patterns. In monkey T(t), beta profile differences were seen during the late delay ($PG > SG$) in both planar and synchronized occurrence probabilities. In monkeys L and T(t), planar pattern occurrence probabilities differed to a small degree before reward onset ($SG > PG$), and in monkey T(t) also the probability of observing synchronized states. Finally in monkey N(i), a small difference in beta power during the early cue ($PG > SG$) was reflected in the probability to observe planar patterns.

4.3.4 Quantification of phase patterns

So far our findings revealed that the modulation of the probability to observe any of the five phase patterns, both in time and by the behavioural condition, was correlated with the average beta power. This suggests that the spatial organization of oscillatory activity, represented by the different phase patterns, may not only be reflected in the trial-averaged power in a statistical sense, but that indeed power modulations and their potential underlying spindle formations correlate with the pattern activity on a single trial level.

As a first step to understand the properties of phase patterns in single trials, we quantified features extracted from the classification results. As classification was performed on single time points, we first calculated the durations of epochs of consecutive time points being classified as the same pattern. In Figure 4.8, we show in A the resulting distributions of the durations for each of the pattern types.

Naturally, these statistics depended on how conservative the choice of thresholds for pattern detection was set. However, given that thresholds were set in accordance to the visually observed phase pattern (cf. Figure 4.3), they served as a visually inspired characterization of the observed sequence of patterns. We found that, on average, planar, synchronized, and radial patterns all had longer durations than random and circular patterns (see large dots in A of Figure 4.8), all in the order of less than one cycle of the beta oscillation ($\approx 40-50\,ms$).

In a next step we examined the preferred direction of the phase gradients of the wave patterns. Here we only considered planar phase patterns (cf. B in Figure 4.8) for which the measure was equivalent to the direction of movement of the planar wave front. Planar waves in monkeys L and T(t) were preferentially observed in the anterior-medial to posterior-lateral direction (see inset for cortical space), whereas waves in monkey N(i) were observed in the anterior-lateral to posterior-medial direction. Note that the array location in monkey N(i) differs from that in monkeys L and T(t), cf. Section 2.2.2, as well as Figure 2.2 and Figure 2.3. This observation is compatible with that described in (Rubino et al., 2006). Even though it was possible to calculate the direction of the phase gradients of any

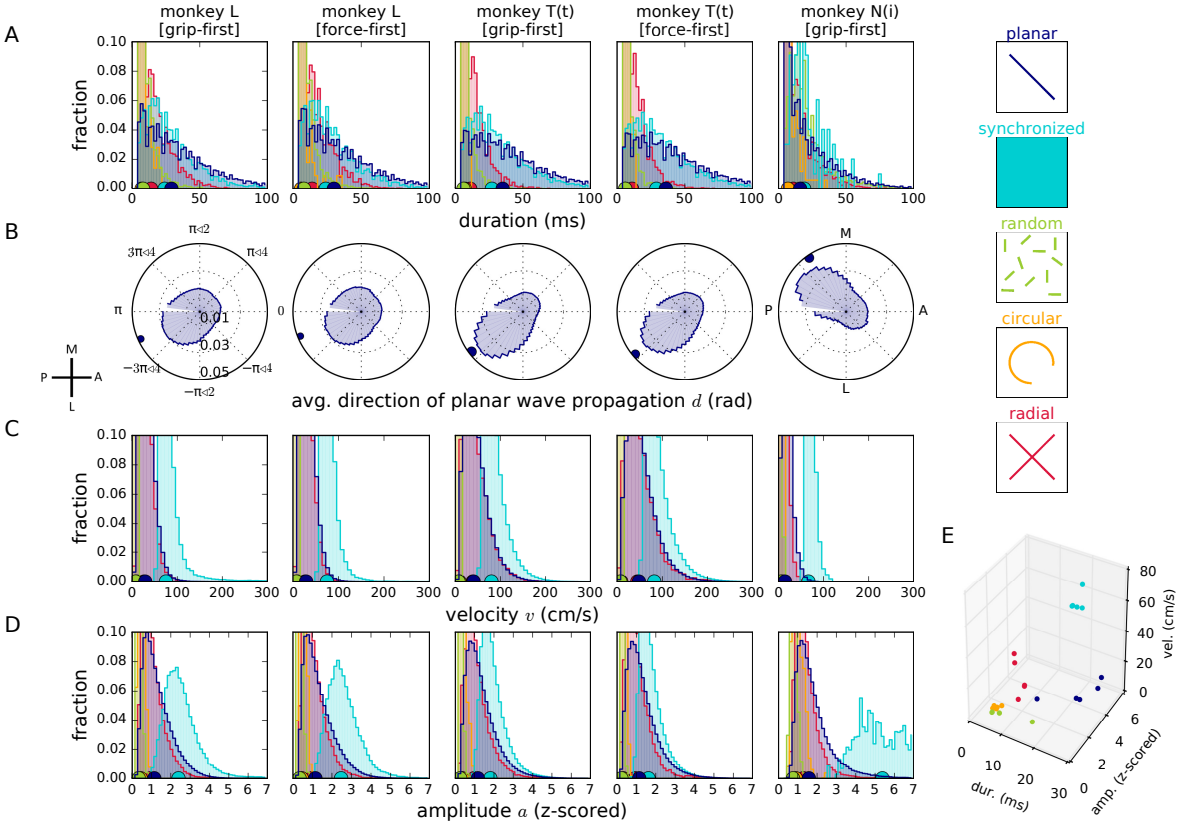

Figure 4.8 – Statistics of detected phase patterns. (based on figure 5 of the unpublished manuscript Denker et al., prep) **A)** Histogram of durations of epochs of consecutive time points classified as belonging to the same pattern (cf. A of Figure 4.7). **B-D)** Distributions of the direction $d(t)$ (panel **B**), phase velocity $v(t)$ (panel **C**), and amplitude $a(t)$ (panel **D**), as a function of the detected phase pattern. Data are separated (columns) according to monkey and recording condition (grip-first vs. force-first). Histograms for different phase patterns are plotted overlapping in the colour corresponding to the legend on the right. For each phase pattern, a histogram entry in panels B-D represents the measured quantity averaged across the array calculated at a time point classified as that pattern. In panel B, the average direction of the phase gradient is plotted in brain coordinates by rotating the activity, and mirroring data along the medio-lateral axis for monkey T(t) to compensate for the array placement in the opposite hemisphere as compared to L and N(i). Large semi-circles: medians of the corresponding distributions. **E)** Joint representation of the medians of the distributions shown in panels A, C, and D. Each data point represents the median of one monkey in one recording condition for one pattern class (indicated by colour).

wave pattern, we refrained from showing the distribution of directions for patterns other than planar patterns, since their characteristics do not allow a clear interpretation of wave propagation direction.

To investigate the dynamical aspect of the wave propagation, we calculated the average wave velocity $v(t)$ (cf. Section 4.2.2) at each time point (cf. C of Figure 4.8). For planar wave patterns, this was directly interpretable as the propagation velocity of the observed wave front. The median propagation velocities of the planar waves were $v(t) = 28.95 \pm 10.19\,cm/s$ (grip-first) and $v(t) = 28.98 \pm 10.29\,cm/s$ for monkey L, $v(t) = 40.53 \pm 16.16\,cm/s$ (grip-first) and $v(t) = 40.53 \pm 16.16\,cm/s$ (force-first) for monkey T(t), and $v(t) = 15.50 \pm 4.72\,cm/s$ (grip-first) for monkey N(i) (all values: median ± median absolute deviation). These values are in rough agreement with those reported in (Rubino et al., 2006). For the other wave patterns, even though it was possible to calculate a velocity, it may not be directly interpreted as the velocity of a propagating planar wave front since phase gradients do not align across the array. Instead, it is a measure that simply captures the average velocity calculated from the local velocities across the array. Synchronized patterns could be considered as a special case of planar waves with a very large spatial wavelength, and as a consequence they exhibited high (in theory, infinitely high) velocities (cf. C in Figure 4.8). On the other hand, random and circular patterns were characterized by phase values that differed strongly between adjacent recording sites.

Therefore, their average velocities were rather low. Finally, radial patterns resembled the planar patterns in that they could be approximated by a planar wave front at a large distance from the centre of the radial pattern. In agreement with this interpretation, they exhibited similar phase velocities as observed for the planar pattern. The five distinct wave patterns showed clear differences in the distribution of their velocities, where a low $v(t)$ corresponds to random or circular patterns, a medium $v(t)$ relates to planar or radial patterns, and a high $v(t)$ indicates the presence of a synchronized pattern. In this sense, the value of the velocity represents a reliable proxy that informs us about the quality of the observed pattern.

After having quantified the features of the wave patterns in single trials, we come back to the question of how the beta amplitude relates to the occurrence of a wave pattern. In Figure 4.8, we show in D for each monkey and behavioural condition the distribution of amplitude components $a(t) = a_i(t)$ across electrodes during each of the five phase patterns. As predicted in Section 4.3.3 from trail-amplitude averaged data, we observed a clear relationship between the instantaneous strength of the beta oscillation, i.e., the amplitude of the envelope of the filtered signal, and the wave pattern. Circular and random patterns occurred at small amplitudes, planar and radial patterns at intermediate amplitudes, and only synchronized patterns occurred at high amplitudes. Therefore, the approximate correspondence between the probability of observing a pattern and beta power seen in the trial average seems to be reflected in the relationship between the single-trial amplitude modulation of the beta oscillation, reflecting the occurrence of spindles, and its spatial organization that is calculated on the basis of phase information.

In Figure 4.8, the results shown in A, C and D were summarized in E where for each wave pattern, monkey and behavioural condition the averaged data for duration, velocity and amplitude are plotted against each other. This figure clearly shows a clustering of collective data points for each individual wave pattern. Thus, the five phase pattern classes are characterized not only by the measures used for pattern classification (cf. C of Figure 4.6), but also by a specific combination of duration, velocity, and amplitude.

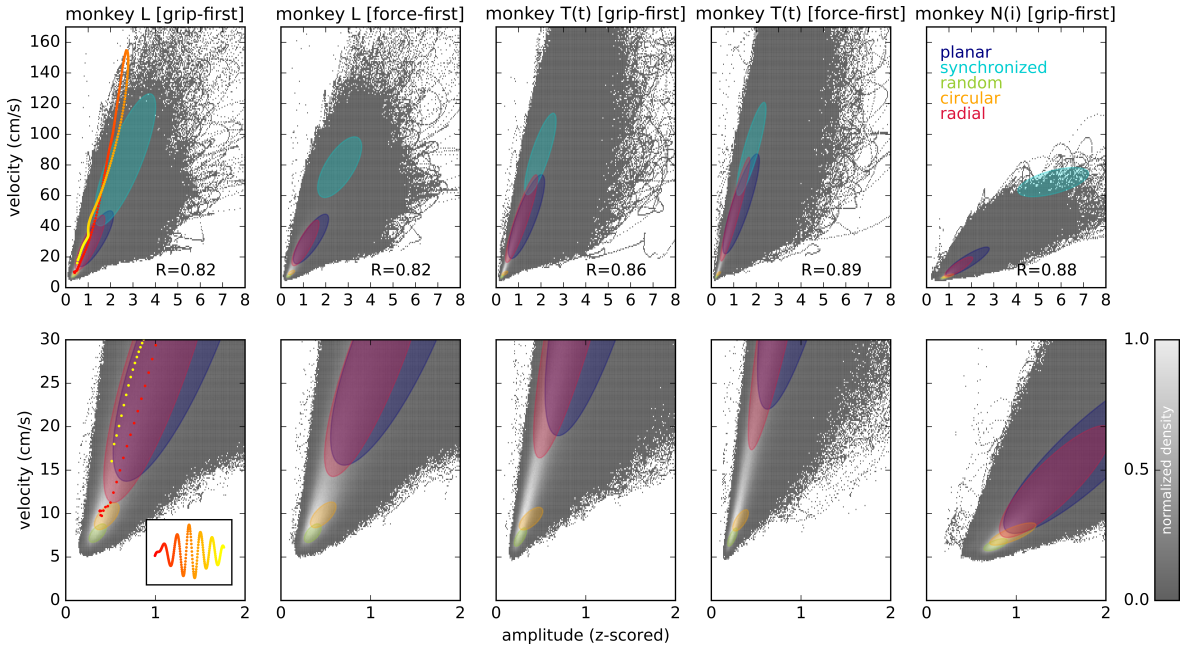

Figure 4.9 – Correlation of instantaneous beta power and spatial pattern of phases, quantified by the phase velocity. (based on figure 6 of the unpublished manuscript Denker et al., prep) Upper row: 2-D histograms of phase velocity $v(t)$ and amplitude $a(t)$ evaluated at each time point (independent of the detected phase pattern) shown (in columns) for each monkey and behavioural condition. Grey values indicate density of time points falling in each histogram bin, normalized to the largest entry of the histogram. The values of the Pearson correlation coefficients R are given in the bottom right of each panel. Each ellipse represents the distribution of time points classified as a specific phase pattern (indicated by colour). Centre of ellipses: mean; radii of ellipses are given as standard deviation in the direction of the two principle components. Lower row: zoomed-in versions of the upper histograms. Left column: Inset in lower panel and dots in red to yellow shades: Illustration of spindle dynamics by example of the spindle after CUE-OFF presented in A of Figure 4.6. Inset reproduces this spindle (transition from red to yellow colours indicates increasing time), corresponding detected states are shown as bars above the spindle. For each time point of the spindle, the corresponding values of the amplitude and phase velocity are marked in the histograms using the identical colour.

4.3.5 Beta amplitude determines phase pattern

In the last step of our analysis, we may now ask if the relationship between amplitude and spatial organization holds for any time point independent of whether or not it can be unambiguously attributed to any of the idealized phase patterns. In order to obtain such a time-resolved view of how the amplitude (which in itself did not exhibit a strong spatial organization, cf. D of Figure 4.5). correlates with the temporal evolution of the patterns, we employed the phase velocity $v(t)$ as a proxy for the spatial organization.

In Figure 4.9 we show the correlation between the instantaneous beta amplitude $a(t)$ and the phase velocity $v(t)$ for each time point for all three monkeys, independent of the phase pattern classification, thus including instances during which no pattern could be classified by our conservative classification algorithm. We observed that the two variables were positively correlated and correlations were highly significant ($p < 0.001$). Thus, an increase in amplitude goes along with an increase in phase velocity, meaning that higher amplitudes led to patterns where neighbouring electrodes were more similar in phase (cf. also C in Figure 4.8). As shown in Section 4.3.4, the velocity $v(t)$ is indeed a good correlate of the perceived organization of beta activity on the electrode array. To more directly illustrate how this relates to the previously defined phase patterns, we indicate in Figure 4.9 by ellipses the regions of the histograms where the individual classes of phase patterns are predominantly found.

Combining these findings, we conclude that observing the temporal modulation of the amplitude of beta activity on a single recording electrode, which relates to the observation of beta spindles, is sufficient to predict the type of spatial organization of beta activity: high instantaneous LFP beta power corresponds to large amplitudes $a(t)$, or to the peaks of spindles, whereas intermediate values of $a(t)$ are associated with the rising and falling flanks of the beta spindles, or the peaks of spindles that exhibit low power (see E in Figure 4.8). For illustration, we visualized the temporal evolution of one spindle and its pattern classification in the left column of Figure 4.9 and observed the corresponding smooth trajectory in the space of amplitude and phase velocity. Therefore, we hypothesize that the modulation of spindle activity goes along with wave-like activity (planar or radial) that progressively increases in speed as the LFP beta amplitude increases. This strong correlation between amplitude and velocity suggests that at the spindle peak also the velocity peaks, which, for large spindles, corresponds to large spatial wavelengths that are perceived as synchronized states on the spatial scale of a Utah-Array. In contrast, a low LFP beta power indicates the time between spindle occurrence and goes along with random or circular patterns at low $v(t)$.

4.4 Discussion of the research results

Three main goals guided this work (cf. Section 4.1). First, we aimed to obtain a more complete description of the wave-like spatio-temporal activity patterns exhibited in the LFP beta range in monkey motor cortex during a complex delayed motor task, and thereby exceeding reports of the planar wave propagation (Rubino et al., 2006; Takahashi et al., 2015). Second, we aimed to relate the wave patterns to behaviour to determine their possible functional implications. And third, we asked in how far these patterns, determined solely by the phase of the oscillation, are related to the instantaneous modulation of the beta amplitude, termed spindles.

Motor cortex beta oscillations exhibit a variety of spatio-temporal patterns By inspection of the dynamics of LFP activity across the electrode array, we demonstrated that beta oscillatory activity shows a number of salient types of spatio-temporal patterns, in addition to travelling planar

waves, such as quasi-synchronized, random, radial, and circular patterns. We developed a phenomenological classification method that identifies epochs that unambiguously exhibit one of the 5 pattern classes. Our approach detected those in a very conservative manner in order to capture the qualitatively salient patterns that are also identified by a human observer. Indeed, the algorithm tends to leave a large amount of time points unclassified, due to the difficulty to clearly attribute a pattern to one of the five idealized pattern types. The reason for this is twofold: On the one side, the coarse-grained resolution of the Utah-Array and the intrinsic variability of the data provided only rough estimates of the phase gradients. On the other side, the patterns were sometimes indeed ambiguous. Often, planar wave fronts were not completely planar, but showed a slight curvature, which could be also interpreted as radial waves. Radial patterns were not necessarily centred on the array, complicating their detection. Random states often exhibited a slight degree of correlation between activities recorded on neighbouring electrodes, contradicting the *a priori* assumption of pure independence.

By using the phase velocity as an easily accessible measure to quantify wave states even for patterns that were difficult to classify, we showed that the distributions of velocities were representative of specific wave states (see C of Figure 4.8). This is not a surprise, since the velocity vector is tightly coupled to the arrangement of phase gradients across the array (see E of Figure 4.8). Thus, we were able to analyse not only time points during which clear patterns were observed, but considered all other time points as well in order to gain a complete picture of the time course of pattern progression as beta amplitude was modulated. However, the instability of pattern types may lead to the interpretation that some of the salient pattern types were dynamically identical: radial patterns may have appeared nearly planar wave-like some distance from the array centre, and quasi-synchronized states appeared in the limit of planar waves approaching infinite phase velocity. This similarity of wave patterns was also reflected in the statistics describing the occurrence of the patterns, e.g., the duration of radial and planar patterns.

Different wave patterns occur at different times during movement preparation and execution The probability of detecting a specific wave pattern was variable during trial of our reach-to-grasp task. Planar and synchronized patterns occurred more often during the pre-cue epoch and during the delay whereas random and radial patterns were more likely to occur around movement execution (cf. Figure 4.7). This observation is in agreement with the hypothesis that planar and synchronized patterns could be triggered by the arrival of visual information in the motor cortex (Takahashi et al., 2015). In line with this view, the orientation of planar wave propagation in our data is in agreement with previous studies (Rubino et al., 2006). More precisely, we found direction preferentially aligned along the antero-posterior axis. The orientation was more anterior-medial to posterior-lateral in monkeys L and T(t), whereas in monkey N(i) it pointed from anterior-lateral to posterior-medial. This difference could reflect the fact that the array was implanted more medially in monkeys L and T(t) than in monkey N(i) (cf. Figure 2.3 and Figure 2.2). Therefore, it seems that all the waves travel toward a medial point along the central sulcus, probably at the level of the hand and finger representation (”nested organization”, Kwan et al., 1978; ”horseshoe” structure, Park et al., 2001). This directional preference may be structured by the underlying connectivity of this cortical area (Kwan et al., 1978).

The predominance of random and radial patterns during movement execution suggests that the spatio-temporal dynamics of neuronal activity is strongly altered during this phase. This could reflect the fact that during movement, information processing is more local and activity propagation is spatially constrained to the motor cortex. However, this hypothesis can be hardly tested at the spatial scale of a single Utah-Array. Multiple Utah-Arrays or optical imaging techniques would be required

to measure the neuronal dynamics at the mesoscopic scale across a larger cortical territory (Muller et al., 2014). Interestingly, Figure 4.7 also suggests that across trials, the probability of observing different wave patterns follows the trial-averaged power profile of the beta oscillation. Namely, planar and synchronized waves are present during phases of high beta power whereas random waves are prominent during periods of low power. The relation to power of circular and radial patterns is more ambiguous. This observation would support the hypothesis the wave dynamics is in part driven by the power of beta oscillations.

Wave dynamics relate to the modulation in beta power Figure 4.8 suggests that low beta amplitudes are linked to random or circular activity patterns with low velocities, intermediate amplitudes to planar or radial wave patterns with intermediate velocities, and that the highest amplitudes indeed co-occurred with quasi-synchronous activity expressing by far the highest velocities (see especially E of Figure 4.8). The pattern statistics also show that the time periods during which one particular, clearly structured pattern was observed were typically of very short duration, in the order of 1 or 2 cycles (see A of Figure 4.8). This is reminiscent of the short-lasting high amplitude events of a few cycles of beta oscillations described by others, the so-called spindles (Murthy and Fetz, 1996a,b). Indeed, these observations point to a tight relationship between spindle dynamics and the occurrence of wave activity. This relationship is well illustrated by the correlation plots in Figure 4.9. In all monkeys, we observed that with growing power amplitude, wave propagation tend to accelerate. For high amplitude beta signals, the wave pattern accelerated to such high levels that the observed pattern became synchronous, which in the ideal case would exhibit infinite velocity.

In summary, the formation of a structured, directed pattern, its acceleration to a near-synchronized appearance, followed by deceleration, and its breakup marked the typical relation of the spatio-temporal organization of the formation of a beta spindle, its peak, and decay (left column in Figure 4.9). Additionally, it has been shown that the maxima of such LFP spindles tend to synchronize across electrodes (Murthy and Fetz, 1992, 1996a), which is a necessary condition to form wave patterns. Both observations are in line with the highly dynamic nature of our pattern occurrences.

A possible model that may benefit from such a scaffold for information exchange is the concept of communication-through-coherence, proposed by Fries (2005; 2015). In this framework, the coherence and phase-relationship between oscillations on different electrodes were taken as a measure of the ability of neurons to entrain their spikes to these oscillations (see also Denker et al., 2011) such that they are more susceptible to inputs from other neurons of the correspondingly synchronized population.

This concept becomes intuitive when considering two brain areas oscillating at similar frequencies, but with a distinct phase-lag. However, on the mesoscopic scale, such as our course-grained recordings from a Utah-Array, where the phase lags between electrodes continuously change in addition to the overall pattern of these phase lags, the picture becomes less clear. Nevertheless, we may hypothesize that if activities on different electrodes become increasingly synchronized with decreasing phase lag as spindles increase their amplitude, information can be more easily communicated across the resulting pattern. This indicates that spindles act as a time window for enhanced cortical communication, not just by means of the strength of synchronization within the local population of neurons (beta amplitude), but because this goes along with a wide-spread zero-lag synchronization of the oscillatory activity (synchronized patterns). Indeed, findings of spike sequences that align to the principle direction of phase gradients (Takahashi et al., 2015; Torre et al., 2016) support this view of a functional mechanism that underlies the generation of beta phase patterns.

This line of arguments raises the question how the patterns of wave dynamics are related to synchronization on the level of single neuron spiking activity. Indeed, it has been shown that the spiking activity synchronizes with the oscillatory spindle peaks (Murthy and Fetz, 1996a) and the cross-correlation histograms of the spiking activity of pairs of neurons become oscillatory in the beta range during periods of strong beta activity (Murthy and Fetz, 1996b). Furthermore, the entrainment of single neuron spiking activity to the LFP oscillation increases with its amplitude (Denker et al., 2007). Additionally, we have recently shown (Denker et al., 2011) that at moments of excessive transient spike synchronization in the order of a few milliseconds that exceed the expectation based on firing rate (Riehle, 1997) spikes lock even more strongly to the LFP beta oscillation than expected, based on the phase locking of those spikes which are not involved in such synchronous events. This effect of particularly strong locking of significant spike coincidences was observed especially during high beta amplitudes. Interpreting the occurrence of excess synchrony as reflecting active cell assemblies, we embedded our findings in a theoretical model that predicts that activated cell assemblies are entrained to the LFP oscillation at a specific phase shortly preceding the trough of the oscillation (Denker et al., 2010). Combining these findings, we may speculate that the occurrence of a beta spindle is not only indicative of the spatial pattern of LFP beta activity, as shown in this study, but additionally beta spindles may govern the temporal relationship of spike patterning observed across the array, as shown by (Takahashi et al., 2015; Torre et al., 2016).

In summary, despite the fact that motor cortical beta oscillations show a strong correlation between signals recorded over large distances, the phase relationships are highly correlated to the amplitude modulation of beta activity, which in turn has been related to the dynamics of spike synchronization on the scale of both firing rates and the precise coordination of spikes. Thus, we believe that the investigation of amplitude and wave pattern will provide a novel leverage on understanding the coordination of activity within spiking neuronal network.

Chapter 5

Conclusion

In light of the increasing volume of data generated in complex experiments, neuroscientists, in particular in the field of electrophysiology, are facing the need to improve the workflows of the daily scientific life. In 2011, Dr. M. Denker conducted a survey[1] among members of the electrophysiology and modelling community ($N = 52$) to better understand how scientists think about the current status of their workflow(s), which aspects of their work could be improved, and to what extent these researchers would embrace efforts to improve their workflows (see selected questions A-E with responses in Figure 5.1).

In this survey, nearly 50% of responders reported that the increased complexity of datasets greatly influences their work (see question D in Figure 5.1). When asked about which features characterize their data, it is obvious that multiple factors of complexity come into play, including the number of sessions, data size, and dependencies between different data records (see question A in Figure 5.1). In total, over 90% believed that making available best-practice guidelines and workflow solutions would be beneficial for the community (see question E in Figure 5.1). Handling data and metadata via standardized formats or frameworks represent one option in designing best-practice for building such a workflow (see question C in Figure 5.1). In fact, nearly 70% of the responders believed a common data format and over 40% of the responders believed that a common description of metadata would be required to achieve that scientific work can be reproduced, verified and extended by other researchers. Over 40% of researchers stated also that they find it difficult to compare their results to those obtained by other researchers working on the same or similar data due to differences in preprocessing steps and data selection (see question B in Figure 5.1).

In line with this survey, the complexity of the datasets from the described R2G experiment (high-dimensional, multi-scale during complex behaviour, cf. Chapter 2) created a challenge for data sharing and performing reproducible analysis already within the closed collaboration of the experimentalists of the CoMCo group in Marseille and the theoreticians of the SN group in Germany. The difficulties mainly originated from (i) the sheer number of shared datasets (see Table 2.1), (ii) the rather variable nature of the circumstances under which the datasets were recorded (cf., e.g., Section 2.1), (iii) the necessity to perform, often interactively, a number of technical validations of the data before they were usable in analyses (cf. Section 2.4), (iv) the use of different programming languages (experimentalists preferred MATLAB, while the theoreticians programmed in Python), and (v) the initial lack of a comprehensive metadata collection and a data framework that would have enabled the members of the collaboration to link and access data and metadata. These difficulties, which the collaboration

[1] http://www.csn.fz-juelich.de/survey/

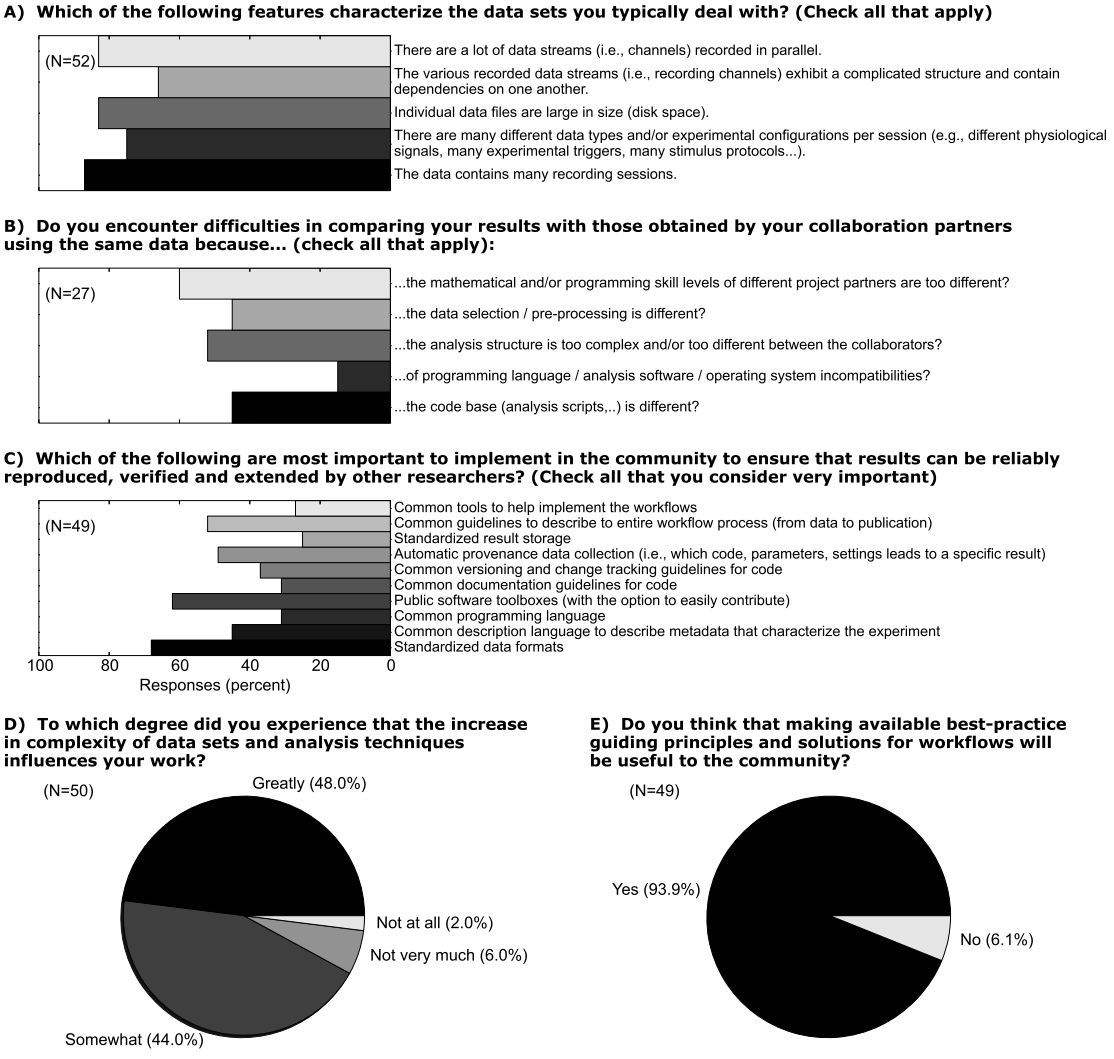

Figure 5.1 – Survey on workflows in electrophysiology - selected questions. (figure 10 of Zehl et al., 2016) Responses to five selected questions taken from a survey among scientists who deal with electrophysiological data sets. The online survey was hosted on "Google Drive" and a total of 52 responders filled the questionnaire in the time period June 12 through October 17, 2011 (Associate/full professor: 18%, Assistant professor: 22%, Post-doc: 28%, Ph.D. student 30%, no response 2%; self-reported). For questions A-C multiple answers could be given by a single person. The labels of the x-axis of response-bar-plot for questions A-C are displayed in C. The number of responders N is provided separately for each question. The complete survey is accissble online via `http://www.csn.fz-juelich.de/survey/`.

partners of the R2G experiment faced in the beginning, are common also for other collaboration projects (cf. in particular panel B in Figure 5.1). This unfortunate circumstance leads to the fact that electrophysiological data provided along with a research publication or as a publicly available data resource are often not well annotated and documented, which reduces above all the chances of reproducible research and the reuse of data by third parties. As I elaborated in Chapter 1, the reason for researchers not to properly address and solve these difficulties originates not necessarily from the negligence of current scientists, but is mostly based on the lack of knowledge how to implement available generic solutions to adequately manage data and metadata (Pulverer, 2015a; Open Science Collaboration, 2015). The work performed during my doctoral studies aimed to change this situation for the members of the research collaboration of the R2G experiment as I will summarize in the following paragraph.

I started by conducting a proper documentation of the R2G experiment in form of a detailed data descriptor following the guidelines of Scientific Data, a state-of-the-art data journal (Chapter 2). This data descriptor, included a profile for all monkeys used for this project (see Section 2.2), a documentation of the task and daily routines the animals had to accomplish with all possible behavioural conditions (see introduction of Chapter 2 and Section 2.1), a full description of the experimental setup including all hardware and software components of the experimental apparatus, the behavioural control system, and the neural recording platform with the Cerebus DAQ system of Blackrock Microsystems (see Section 2.3). In addition, the descriptor contains a summary of required technical validations of the data (see Section 2.4) in form of (pre-analysis) preprocessing steps (e.g., as the offline spike sorting described in Section 2.4.3) or quality assessment procedures (e.g., as the electrode and trial rejection described in Section 2.4.4.3). To be able to provide formalized human and machine-readable metadata along with this formulated documentation, I reorganized the (re)collected metadata and compiled a standardized, comprehensive metadata collection using the odML framework (introduced in Section 1.2). This reorganization reduced the distributed metadata (over several files and formats) of the experiment to one odML-file per recording (Section 2.5.1), which already facilitated the metadata handling for screening or analysis procedures (Section 3.1). I further provided in a tutorial-like style how to prepare and conduct the compilation of a comprehensive odML metadata collection (Section 3.2), and how to utilize an existing odML metadata collection using Python and MATLAB (Section 3.3). In addition, I provided a working solution based on the Python programming language to generically read the primary data in combination with the corresponding metadata of the R2G experiment (Section 2.5.2). For this, I implemented two data loading routines using the Neo data framework (introduced in Section 1.3): the `BlackrockIO` as generic I/O module of the Neo framework for loading primary data originating from any Cerebus DAQ system (Section 2.5.2.1) and the subsequent `ReachGraspIO` as experiment specific I/O module that attaches relevant metadata extracted from the odML metadata collection as annotations to the loaded Neo data objects hosting the corresponding primary data of the R2G experiment (Section 2.5.2.2). In Chapter 4, I finally demonstrated how information of the data descriptor and how the formalized access of data and metadata via the corresponding frameworks are integrated and used to conduct a concrete research analysis. The presented research analysis of in the R2G experiment focuses on the inspection of the dynamics of the beta oscillations ($13-30\,Hz$) of local field potential (LFP) activity across the electrodes of the Utah-Array and demonstrates that depending on the amplitude of these oscillations their phase shifts exhibit different types of spatio-temporal patterns (planar wave, synchronized, random, circular, and radial, cf. Section 4.3.2).

In the following two paragraphs, I will discuss the impact of providing a data descriptor for an experiment and the usage of standardized metadata and data frameworks, such as odML and Neo,

on collaborative work within and between laboratories, the potential to publish original experimental data and on conducting publications of data analysis procedures.

Impact of creating a data descriptor. In Chapter 2, I illustrated the necessary content of a formalized documentation in form of a data descriptor characterizing the R2G experiment based on the guidelines of data journal Scientific Data. The time and effort needed to write such a data descriptor greatly influences the willingness of researchers to share their data with third parties. If one considers the fact that a complex experiment which prolonged over years usually also comprises failures and unsuccessful attempts, as well as changes and developments (e.g., update of hardware and software components), the requirements of a data descriptor are indeed more difficult to fulfil. Moreover, it entails a transparency about the author's workflow which is not given light-mindedly in a competitive scientific field. For the R2G experiment the consequences of such a data publication were therefore carefully discussed, with the result that only two out of over thousand datasets of the example experiment were published. Nonetheless, these two datasets are of high scientific interest for the community (cf. Section 2.6). In addition, the data descriptor not only facilitates the further documentation of the running R2G project, but will also support sharing data between members of the involved collaboration partners (cf. Section 2.6) by providing an explicitly formulated common knowledge space of the experiment. For the research analysis presented in Chapter 4, the data descriptor provided the foundation to describe the subjects, the conducted surgical procedures, the configuration and cortical location of the corresponding arrays, the task paradigms and settings (conditions and trial type sequence) of the selected datasets, and the technical validations performed on the data.

Based on the usability of a formalized documentation of an experiment, I highly recommend researchers to write a data descriptor for every project and, ideally, to start writing it in parallel with a running experiment. The latter allows scientists to distribute the corresponding effort to create a complete data descriptor over time, and avoids the cumbersome process of recollecting the needed information based on distributed notes and memorized knowledge later on, which risks costly loss of data and information.

Impact of using standardized metadata and data frameworks. Based on the guidelines of the data journal Scientific Data, besides the formalized data descriptor, the metadata and data of the experiment has to be provided in machine-readable formats on a publicly accessible repository. In parallel, the journal recommends adding a usage note to the descriptor, illustrating how to access and use data and metadata of the provided formats.

To utilize the metadata contained in the data descriptor of the R2G experiment for automated analysis procedures, I collected and structured the corresponding metadata into a standardized, comprehensive odML metadata collection (cf. Section 2.5.1). The general advantages of such a comprehensive metadata collection are (i) to facilitate maintenance and exchange of metadata (cf. Case 1 in Section 3.1), (ii) to optimize human and machine-accessibility to the metadata by pooling information from various sources (cf. Case 2 and 3 in Section 3.1), (iii) to create textual and graphical representations of sets of related metadata in a fast manner in order to screen information across the experiment (cf. Case 3 in Section 3.1), and (iv) to allow for a simple and well-defined selection of data based on metadata information using standard query mechanisms that formalize communication in collaborations (cf. Case 4 in Section 3.1). For the latter, I integrated the odML metadata framework into a common data representation by implementing loading routines based on the Neo data framework (cf. Section 1.3 and Section 2.5.2).

The resulting standardized metadata-enriched data representation guaranteed uniform access for

all collaboration members using Python and, indirectly also for members preferring MATLAB, by storing data as mat or hdf5-files via the Neo MATLAB or HDF5 I/O modules. With this well selected set of available technologies, I provided a workflow to better coordinate metadata-based data selection and access, which simplifies the implementation of analysis procedures of corresponding research papers across members of the CoMCo/SN research collaboration (sharing level A-C, cf. Section 1.1), as well as the publication of datasets of the R2G experiment for the use by third parties (sharing level D, cf. Section 1.1).

More concretely, for the publication of the two datasets of the R2G experiment, the introduced workflow makes it possible to publish for each dataset a neat collection of files consisting of the original data files of the Cerebus DAQ system (ccf, nev, ns2, and ns5/6) and one odML-file containing all relating metadata (cf. Section 2.6). Moreover, all code necessary to load either metadata and data separately via the open-source libraries of the odML and Neo framework or as metadata-enriched data Neo objects via the R2G specific loading routine, is publicly available and allows Python as well as MATLAB users access to the published data (cf. Section 2.5.2 and Section 2.6). For conducting the research analysis in Chapter 4, this formalized way of loading and selecting data based on chosen metadata criteria facilitates the implemented code for the analysis, and with this makes it easier to reproduce the applied procedures on the exact same data. A concrete example for the research analysis of the R2G experiment presented in Chapter 4, would be the possibility to automatically exclude electrodes, which were identified as noisy during the quality assessment of the corresponding frequency band of the LFP signals, from the analysis procedure, based on their attached metadata annotation (cf. Figure 4.1).

Although the creation of the introduced workflow was time consuming, the implementation of standardized data and metadata frameworks, such as the Neo and odML framework (cf. Section 1.3, Section 1.2, as well as Section 2.5.2 and Chapter 3), are necessary steps towards a workflow that at least provides the chance to include an entire provenance trail for scientific findings (cf. Section 4.2).

Discussing the limitations of the used metadata and data frameworks. While the storage of metadata using the odML framework in combination with the data representation via the Neo data framework represent viable technical solutions in light of improving the overall experimental and analysis workflow, they cannot solve the intrinsic problem of automatically identifying, collecting and pooling the metadata in an experiment, and cannot relieve the user of having to implement quite an amount of code and functionality to integrate these frameworks into the daily experimental routines. All of the necessary steps are non-trivial, and time-consuming by nature. For this reason, the question may arise whether it is worth the effort.

Through the practical illustration in Chapter 3, I hope to have convinced the reader that indeed numerous advantages are associated with a well-maintained comprehensive metadata collection that accompanies an established data framework of an experiment (see previous paragraph for a concrete example). For the individual researcher these can possibly be best summarized by the ease of organizing, searching and selecting datasets based on the metadata information for a research analysis (see previous paragraph for a concrete example). Even more advantages are gained for collaborative work. First, an easily accessible central source for metadata ensures that all researchers access the exact same information. This is particularly important if metadata are difficult to access, because they are captured as hand-written notes or stored in different source files in different locations, that in the worst case require implementation of customized loading routines. Second, the communication between collaborators becomes more precise by the strict use of defined key-value pairs, lowering the probability for unintended confusion when selecting datasets for analysis procedures. Last but

not least, once the metadata collection structure, content and method of creation are defined by the experimenter, metadata entry and compilation will in general run very smoothly, and be less error prone. Moreover, if the compilation of the metadata collection does not run through, one may detect problems that occurred for individual recordings which otherwise would have been passed unnoticed. The same arguments hold for the access and usage of data via generic and robust loading routines of an established data framework. Thus, I argue that there are a number of significant advantages that justify the initial investment necessary to include an automated metadata management in combination with a common data framework as part of the workflow of an electrophysiological experiment. Furthermore, I assert, that we scientists have the obligation to properly document our scientific work in a clear, transparent fashion that enables the highest degree of reproducibility. Data sharing and the reproducibility of research analysis are becoming increasingly hot topics for publishers and funding agencies (Candela et al., 2015; Open Science Collaboration, 2015; Morrison, 2014; Pulverer, 2015a), such that appropriate resources of time and man-power allocated to produce more sophisticated data and metadata management will become a necessity.

While I believe that the odML metadata framework in combination with the Neo data framework represents an important and useful step towards better data and metadata management, it is also clear that the chosen approaches have a number of potential improvements.

For the odML framework, a natural step would be to become a standard file format for the commercially available data acquisition systems, such that their metadata are instantly available for inclusion in the user's metadata collection. In this context, the odML format so far has been adopted in the connectome file format[2], the Relacs[3] data acquisition and stimulation software, and the EEGBase database[4] for EEG/ERP data. Likewise, the availability of interfaces for easy odML export in popular experiment control suites, vendor-specific hardware or generic software, such as LabVIEW[5], would greatly speed up and facilitate the compilation process of a metadata collection. In this context, a repository for popular terminologies and hardware devices has already been initiated by the G-Node[6] , which is open for any extensions by the community. It is worth mentioning that the odML framework by itself is of a general nature and can easily be used in domains of neuroscience other than electrophysiology, or even other fields of science. A most obvious use case would be to store metadata of neuroscientific simulation experiments. The advantage of this particular example is that a common storage mechanism for experimental and simulated data would simplify their comparison, an issue that is bound to become increasingly important for the future of Computational Neuroscience. In general, using a well-defined, machine-readable format for metadata brings the potential for integration of the information across datasets, for example in larger databases or data repositories.

Another step to improve the process of metadata management would be to provide a convenient way to manually enter metadata into a corresponding collection of an experiment. In the case of the odML framework, while the existing odML-Editor enables researchers to fill in odML-files, a number of convenience features would not only reduce the amount of time required to collect the information, but could also provide further incentives to store metadata in the odML format. An example feature could be an editor support for templates such that new files with default values may be created quickly, or the possibility to display the metadata and their organisation structure in the editor in a more flexible way. The odMLtables library, which can transform the hierarchical structure of an odML-file

[2] https://www.nitrc.org/projects/cff/

[3] http://www.relacs.net

[4] https://eegdatabase.kiv.zcu.cz/

[5] http://www.ni.com/labview/d/

[6] http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml

to an editable flat table in the Excel or CSV format, is one current attempt to solve these issues in the odML framework. I initiated the odMLtables project in the course of the R2G experiment, to increase the accessibility of the odML framework for users with little programming knowledge (e.g., newly recruited students). The conversion of the odML format to commonly known spreadsheets facilitated the integration of metadata into the comprehensive collection during the daily laboratory routines of an experiment. Another particularly notable project aimed to improve the manual entries of metadata during an experiment is the odML mobile application (Le Franc et al., 2014) that runs on mobile devices that are easy to carry around in a laboratory situation.

A more challenging problem is to improve the mechanisms that link metadata to data between the chosen frameworks. This issue becomes particularly clear in the context of the current workflow of the R2G experiment. For example, the generic I/O module of the Neo data framework (`BlackrockIO`) reads a particular spike train that relates to a certain unit ID in the recording, while the corresponding odML-file contains for each unit ID the information about the assigned unit type (SUA, MUA, or noise) obtained by the offline spike sorting preprocessing step (cf. Section 2.4.3) and the signal-to-noise ratio (SNR) of each unit resulting from the corresponding quality assessment procedure (cf. Section 2.4.4.4). To combine data with metadata, it was necessary to implement an additional experiment-specific I/O module (`ReachGraspIO`) to annotate the spike train data with this particular piece of metadata. The implementation of such a module is time-consuming, and again may be performed differently by partners in a collaboration, especially when they use a different programming language, which in turn can cause incoherences between workflows and hinders the collaborative work. A better way would be to have a representation in a standardized description language of how data and metadata should be linked, which can be used as a construction manual for loading routines in different programming languages. Another approach would be to immediately store data and metadata in the same format. Recently, the NIX file format[7] was proposed (Stoewer et al., 2014) as a possible solution to link data and metadata on the file level. In this approach metadata are organized hierarchically as in odML, but can be linked to the respective data stored in the same file, which can be loaded with a available generic loading routine. This enables data and metadata to be meaningfully related, in order to facilitate and automate data retrieval and analysis without the need to supply collaborators with experiment-specific code. However, also in this approach, data and metadata need to be first linked together before saving them in a commonly accessible file. With this, not only the necessity to provide routines that integrate experiment specific metadata into the nix-file format and link them to the corresponding data objects remains, but it also requires a data duplication. Furthermore, if datasets of the NIX format are to be synchronized to a different location (e.g., to a server of a collaborating laboratory), even small changes to the metadata could involve a complete resynchronization of an entire dataset.

Finally, an unsolved problem of the current workflow approach of the R2G experiment is the implementation of an automated provenance trail for the scientific findings of the conducted research analysis procedures. Tools for recording provenance trails provide the semantic context in which scientific findings were produced, which is the next crucial step towards reproducible research (Davidson and Freire, 2008). For the reproducibility of research, making metadata, data and code available to others is not sufficient on its own. In addition, by using the available metadata, it is necessary to also provide a report of the correct semantic order of what code was run with which data parts, parameters and in what computing environment to generate a given result, e.g., presented in a figure of a research paper (Brammer et al., 2011; Koop et al., 2011; Nowakowski et al., 2011). Corresponding software should automatically record this provenance trail for scientific findings. In neuroscience, due to the

[7] http://www.g-node.org/nix

heterogeneity of data and metadata formats and the resulting variety of (often home-brewed) software solutions to access and analyse them, software solutions for provenance trails are not yet established, especially not for such complex electrophysiological experiments as the one presented here. Current approaches from neuroinformatics/computer science are unfortunately not yet sufficient to generate a complete provenance track of the correspondingly scrappy processes (e.g., Badia et al., 2015 or Curcin and Ghanem, 2008). For this reason, the procedures of the example research analysis of the R2G experiment presented in Chapter 4 are not yet completely provenance tracked. Nevertheless, the usage of the Neo data and odML metadata framework provided consistent and reproducible data selection and access. It further provides the use of generalized analysis routines provided by a common data analysis software tool, called Elephant[8] (cf. Section 1.3). As a consequence, it was possible to store additionally to the results a set of hdf5-files (one for each applied analysis procedure) using the standardized Neo HDF5 I/O module that contained provenance information, such as the chosen formalized metadata selection criteria from the odML-files and the parameter settings of the applied analysis functions. Analysis dependencies were guaranteed by the fact that metadata and parameter settings, which were determined for more than one analysis procedure, were stored when they were first used, and then recalled from the corresponding hdf5-file for the consecutive analysis steps. A small improvement to this approach would have been to include also the Git hash code of the version of the analysis scripts in the corresponding hdf5-files, because all used and implemented code was version controlled using Git[9](Chacon and Straub, 2014). In fact, it is the plan to improve the analysis workflow of the research publication by introducing sophisticated, up-to-date techniques for automated managing and tracking of numerical data analysis procedures, such as Sumatra[10] Davison, 2012, Snakemake[11] (Köster and Rahmann, 2012), Taverna[12] (Wolstencroft et al., 2013), or VisTrails[13] (Bavoil et al., 2005), and data dependencies in a panellized data analysis pipeline, such as PyCOMPSs[14] (Tejedor et al., 2015) or WINGS[15] (Gil et al., 2011).

In summary, I demonstrate with my thesis that managing electrophysiological data and metadata to properly share research data within a scientific collaboration is cumbersome and time consuming, but feasible and essential for successfully publishing data and analysis results for a broader audience of users. To promote data sharing within the neuroscientific community and to provide a better foundation for reproducible research, my thesis offers the first documented approach that puts together and extends well-selected, state-of-the-art technologies to form a coherent strategy for data and metadata management for electrophysiological experiments, from initial data agglomeration to a final research project. Furthermore, the suggested workflow, which completes and optimizes metadata-based selection and analysis of research data, also has the potential to foster new scientific insight on highly complex experimental data that are obscure without the corresponding metadata annotations. In my opinion, the complexity of current neuroscientific experiments forces us scientists to reorganize our workflow of data handling, including metadata management, to ensure reproducibility in research (Stodden et al., 2014). The readily available tools to support metadata management, such as odML, and data management, such as Neo, are a vital components in constructing such workflows. It is our

[8] http://neuralensemble.org/elephant/

[9] https://git-scm.com/

[10] http://neuralensemble.org/sumatra/

[11] https://bitbucket.org/snakemake/snakemake/wiki/Home

[12] http://www.taverna.org.uk/

[13] https://www.vistrails.org/index.php/Main_Page

[14] https://omictools.com/pycompss-tool

[15] http://www.wings-workflows.org/

responsibility to propagate and incorporate these tools into our daily routines in order to improve workflows through the principle of co-design between scientists and software engineers.

5.1 Outlook - Paving the way for new scientific findings

The presented infrastructure of metadata, data and workflow management in the R2G experiment, enabled us (cf. contributing authors in List of own papers with authors contributions) to better coordinate the analysis of the presented research publication in Chapter 4. For this infrastructure, I combined a comprehensive and formalized representation of metadata via the odML framework with the common data representation of the Neo framework that enabled us to make use of the common analysis toolbox Elephant, which facilitated the implementation of the analysis methods described in Section 4.2. In short, the implemented infrastructure optimizes metadata-based selection and analysis of research data.

This raises the question of whether this improved infrastructure effects also the development of new scientific insight on the data of the R2G experiment? To answer this question, I will first recapitulate and discuss the background and outcome of the presented research publication and present future plans for this project:

On the level of the local field potential (LFP), beta oscillations ($15-35\,Hz$) recorded on separate electrodes in motor cortex are often highly correlated, but exhibit a non-zero phase shift. These shifts have been shown to organize spatially in the form of planar wave propagation along preferred directions across the cortical surface during an instructed-delay reaching task (e.g., Rubino et al., 2006). However, little has been reported about the spatial organization of beta oscillations outside epochs that exhibit a clear planar wave (for further background see Section 4.1). As result, the presented research analysis was guided by *three main goals*, for which the presented results enabled us to draw the following conclusions (in summary, for details see Section 4.4):

As first goal, we aimed to obtain a more complete description of the wave-like spatio-temporal activity patterns exhibited in the LFP beta range in monkey motor cortex during a complex delayed motor task, and thereby exceeding reports of the planar wave propagation (Rubino et al., 2006; Takahashi et al., 2015). Using the presented approach to classify spatial phase patterns (cf. Section 4.2.4), we were indeed able to demonstrate that, across the electrode array, the dynamics of the LFP beta oscillations between 13 and $30\,Hz$ showed a number of salient types of spatio-temporal patterns, in addition to travelling planar waves, such as quasi-synchronized, random, radial, and circular patterns (cf. Section 4.3.2 and Figure 4.6).

As second goal, we aimed to relate the wave patterns to the behavioural context of the R2G experiment to determine whether the detected pattern class has possible functional implications. In line with observations from previous studies (Rubino et al., 2006; Takahashi et al., 2015), we identified that planar and quasi-synchronized patterns occurred more often during the pre-cue epoch and during the delay (cf. panel B in Figure 4.7) and that the direction of planar wave propagation is preferentially aligned to the antero-posterior axis (cf. panel B in Figure 4.8). This directional preference may be structured by the underlying connectivity of this cortical area (Kwan et al., 1978; Rubino et al., 2006). During movement execution random and radial patterns are predominant (cf. panel B in Figure 4.7), which suggests that the spatio-temporal dynamics of neuronal activity is strongly altered during this phase. This could reflect the fact that during movement, information processing is more local and activity propagation is spatially constrained to the motor cortex, although the spatial scale of a single Utah-Array is too small to actually test this hypothesis.

As third third goal, we asked in how far these patterns, which were determined solely by the phase of the oscillation, are related to the instantaneous modulation of the beta amplitude, which are indicative of observing spindles. Indeed, the results presented in Figure 4.7 (cf. panel B and C) suggest that planar and synchronized wave patterns are present during phases of high beta power whereas random wave patterns are prominent during periods of low power. Furthermore, the statistical analysis of the patterns revealed that low beta amplitudes are linked to random or circular activity patterns with low velocities, intermediate amplitudes to planar or radial wave patterns with intermediate velocities, and the highest amplitudes to quasi-synchronous activity with the highest velocities (see especially E in Figure 4.8). The pattern statistics also showed that one particular, clearly structured pattern was typically observed during short-lasting high amplitude events, in the order of a few cycles of beta oscillations (see A of Figure 4.8), the so-called spindles (Murthy and Fetz, 1996a,b). The maxima of such beta spindles tend to synchronize across electrodes (Murthy and Fetz, 1996a,b) leading to an acceleration of the wave propagation to such high levels that the observed pattern became synchronous (cf. correlation plots in Figure 4.9). Based on the concept of communication-through-coherence (cf. Fries (2005; 2015), and Denker et al., 2011), we hypothesize that if activities on different electrodes become increasingly synchronized with decreasing phase lag as spindles increase their amplitude, beta spindles act as a time window for enhanced cortical communication. This functional mechanism underlying the generation of beta phase patterns is indeed supported by studies demonstrating that (i) spiking activity synchronizes with the oscillatory spindle peaks (Murthy and Fetz, 1996a,b), (ii) cross-correlation histograms of the spiking activity of pairs of neurons become oscillatory in the beta range during periods of strong beta activity (Murthy and Fetz, 1996b), (iii) the entrainment of single neuron spiking activity to the LFP oscillation increases with its amplitude (Denker et al., 2007), and (iv) spike sequences align to the principle direction of phase gradients (Takahashi et al., 2015; Torre et al., 2016).

Combining these findings, we speculated that the occurrence of a beta spindle in the range of $13-30\,Hz$ is indicative of the spatial wave pattern of the LFP beta activity, which may in turn reflect the spatio-temporal dynamics of neuronal activity and may govern the temporal relationship of spike patterning observed across the array. One future study to test this hypothesis for the data of the R2G experiment could directly relate the results of the wave pattern analysis presented in Chapter 4 (Denker et al., prep) with the results of the synchronous spike pattern analysis presented in Torre et al. (2016). For this, both analytical approaches need to be applied to the exact same datasets. In particular, it is important to select the same electrodes and the same trials for the analysis, for which one needs to combine metadata from the quality assessment procedures of the LFP and the spike data (cf. Section 2.4.4). The fact that the implemented odML metadata and Neo data frameworks facilitate formalized metadata-based data selection and provide the opportunity to make use of commonly implemented analysis routines of the Elephant toolbox, will improve the workflow management of such a prospective research analysis and will make it easier to relate the corresponding results to previous studies, which used the same frameworks.

When considering the possibility that wave patterns play an active role in information processing in the brain, one open question raised by our results is to which extent specific features of the pattern indeed carry information. Indeed, the authors of Rubino et al. (2006) have shown a correlation between the observation of planar waves and the direction of an upcoming movement. In contrast, our findings only weakly support a view where the observation of particular types wave patterns correlate with the information given to the monkey about the upcoming movement.

As described in Section 4.3.3, to this end, we only compared results obtained during SG and PG trials of the standard task paradigm (cf. Section 2.1.1) with condition setting 1 (cf. Table 2.2) and randomly alternating trial types. The probability to observe planar and quasi-synchronized waves was (significantly) larger in PG than in SG trials during the early or late delay for monkey L, and monkey T(t), respectively (cf. panel B of Figure 4.7). In contrast, for monkey N(i) only very few synchronized patterns were detected during a trial in general and the probability to observe planar waves was (significantly) larger in PG than in SG trials during the grip cue presentation (cf. panel B of Figure 4.7). In general, we noticed that the wave pattern activity seen for monkey N(i) was far less structured than of the other two monkeys. Whether this is due to the difference in array placement (cf. Section 2.2.2) or differences in the behavioural performance of the monkey is not clear (cf. Section 2.2.1). Nonetheless, our theory that the occurrence of a beta spindle is indicative of the spatial wave pattern of the LFP beta activity was supported by the fact that for all three monkeys the occurrence probability of the planar phase pattern closely followed the time-resolved modulation of beta amplitudes, which is in turn reflecting the probability of beta spindle occurrence in time (beta-profiles in panel C of Figure 4.7). Unfortunately, this tight relationship between beta spindle dynamics (beta profiles) and the occurrence of wave activity also weakens the significance of the deviating probability between PG and SG trials to observe planar and quasi-synchronized waves, because the amplitude difference between the beta profiles of the pre-movement period of PG and SG trials is highly variable between datasets.

The relationship of beta power and behaviour was investigated in a preliminary analysis project of the R2G experiment that I performed during my doctoral studies. In this project, I focused on the beta-profile differences between trials of different grip and force type and tried to determine why the beta profiles sometimes clearly deviates between grip types and sometimes not. The results were inconclusive so far, but I and Dr. T. Brochier found indications that the engagement of a monkey in the task and its resulting performance, as well as the behavioural context, in which a recording or even a trial took place, can have an influence on or correlate with the beta amplitude modulations. Based on these indications I would like to briefly outline some future studies that are needed to further investigate the functional meaning of the beta profile modulations in the R2G experiment.

Recently, the authors of Hosaka et al. (2015) demonstrated for bi-manual sequential motor task with two Japanese macaque monkeys, that in trials where the monkeys had to maintain a certain motor behaviour, the power of the beta-oscillations ($10-40\,Hz$) recorded from the supplementary motor area were persistently enhanced during the early period of an instructed preparatory delay before the requested movement (cf. delay period Section 2.1.1). In contrast, in trials where the monkeys had to change to a different motor behaviour, the power of the beta-oscillations were suppressed. Based on these findings, we could test for the random trial type sequence weather an accidental repetition of the same trial types affects the beta profiles of the corresponding trials, and, with this, the variability of deviations between PG and SG trials. In this context, we could further examine whether we find the same variability of beta profiles for the grip types in recordings where the trial types alternate between blocks of three or more trials, and compare these results of predictable trial type repetition (or the possibility to plan an alternation of the motor plan) to the results of accidental trial type repetition (or the impact of an surprising alternation of the motor plan). Also inspired by the findings of Hosaka et al. (2015), we further could investigate whether a repetition of the same grip condition setting over several recording days (e.g., only conditions with SG trials, cf. Table 2.2) influences the deviation of beta profiles between PG and SG trials, especially for trials of the recording where the condition setting is first changed again. In addition, the authors of Hosaka et al. (2015) report that the suppression of beta power when changing the motor plan is accompanied by an increase in power of

high-gamma oscillations ($40-200\,Hz$), and that a suppression of beta power in a trial sequence where the motor plan needs to be maintained resulted in an erroneous choice of action in the next movement. From these and previous findings (Nakajima et al., 2013), the authors speculate that enhanced beta power may play a role in protecting cortical cell assemblies, which are involved for the current motor plan. Furthermore, they hypothesize that the suppression of beta power affords permission to form new cell assemblies for a new motor plan based on an update of the external information, which is reflected by the increase of the high-gamma power. Based on these theories, we could investigate whether in grip-error trials the beta power is low during the preparatory delay period, whether we can identify the same correlation of high-gamma power, and if spike patterns or the possibilities to identify them (cf. Torre et al., 2016) alternate depending on the beta power.

In a preliminary study that I presented as a poster at Society for Neuroscience conference 2014 (Zehl et al., 2014a), I was able to demonstrate that the timing of the beta amplitude peak and strong subsequent amplitude decrease during the delay period changes in relation with the reaction time (RT) only if the grip type is known (cnd1) and not if the force type is known (cnd2). From this, I concluded that only when the grip type is known the corresponding movement is prepared in advance, and this is reflected by an earlier beta amplitude modulation in the delay period. I further showed that the amplitude deviation of beta profiles during the delay period is more present between PG and SG trials if the grip type is known (cnd1), than between LF and HF trials if the force type is known (cnd2), especially for trials with median or fast RT. If the RT is taken as a measure of how well a monkey was prepared for the upcoming movement, I speculated that the difference in amplitude during the delay period reflected the preparatory differences for the different grip types. Nonetheless, although I could detect dependencies of the beta amplitude modulation with the RTs, the trial-by-trial variability remains high. If one assumes, though, that the beta amplitude modulations reflect how well a monkey is prepared for the upcoming movement, a high trial-by-trial variability is to be expected, because the individual modulations will reflect the trial-specific state of the monkey's attention. For future analysis, it is therefore necessary to focus on (i) identifying metadata that describe this trial-specific state of the monkey's attention, and (ii) performing a trial-by-trial analysis to relate the monkeys behaviour in time directly with the corresponding beta profile modulations. Given the results of Chapter 4 (Denker et al., prep), detailed knowledge about the relationship of beta power to behaviour could provide a more concrete hypothesis on how the spatio-temporal activity patterns in the LFP are functionally related to movement preparation in the motor cortex. For more information on the corresponding preliminary study see Section "Preliminary study on beta profiles" in the Appendix.

For investigating all these possible modulators of motor cortical beta profiles in more detail, the completion of a standardized odML metadata collection, which can be linked to the corresponding data of the R2G experiment, was an important step for (i) performing corresponding metadata analysis to identify potential modulators, and (ii) being able to select data from whole recordings to single trials based on these identified metadata criteria. At the time point of the preliminary study the implementation of the odML metadata and Neo data framework were unfortunately not completed yet, impeding the corresponding analyses. Future studies which involve metadata-based data selection will profit from the now settled metadata and data management of the R2G experiment.

Another aspect that aggravated further analysis of the motor cortical beta profiles is the lack of datasets to compare certain behavioural conditions. In addition, the general character/performance differences between the monkeys (cf. overeagerness of monkey L compared to monkey T and N in Section 2.2.1), and especially the different anatomical position of the arrays (cf. array placement in monkey N(i) compared to monkey T and L in Section 2.2.2) making statistical comparisons and

validations complicated. For this reason, the upcoming datasets from the currently trained monkey E (cf. introduction of Chapter 2) are important to complement the running and future studies on the data of the R2G experiment.

5.2 Outlook - Aiming towards reproducible neuroscience

Our responsibility as scientists is to ensure the reproducibility of our research, to promote the research progress by sharing our research findings and, ideally, also our research data, as well as to propagate and incorporate corresponding up-to-date management and analysis tools, actually forces us to adapt our workflows accordingly. Yet, there is still a general failure in the field of neuroscience to implement data and metadata, as well as automated workflow management solutions (Denker and Grün, 2015).

The reason for this failure is simple, and actually obvious through the content of this thesis, because the practical illustrations I gave for implementing a data and metadata management for a neuroscientific experiment using up-to-date techniques remain highly complex, time-consuming and laborious even if the outcome drastically reduces the later investment for (re)using the data and metadata. Further practical guidance for integrating automated workflow management tools, such as Sumatra, are not yet included, but are necessary for a complete and robust provenance trail of research analyses. Especially experimentalists, who have little time to learn specifics of new data, metadata, and workflow standards (Tenopir et al., 2011), are further inhibited by the current programming expenditure the usage of these standards still entails, because often an intensive programming training is not part of their education.

This leads us back to the necessity to further improve scientific workflows through the principle of co-design between scientists, and in particular experimentalists, and software engineers (keyword: share and combine different knowledge space) to avoid the "*design* (of) *complex systems for capturing comprehensive descriptions of experiments that are time-consuming to populate and thus never used*" (Paton, 2008). Based on the experience I gained from my doctoral studies, I would first aim my effort on the following improvements for data, metadata and workflow management:

Facilitation of collecting metadata Metadata frameworks, such as odML, are often based on the development and optimization of the tool along a specific use case that is available to the responsible software engineers. As result, the software tools are highly sophisticated in their original context, but lack general applicability to other projects and often become quite difficult to handle in a different context. For example, the odML framework was originally developed to automatically manage metadata of the DAQ system Relacs. Using the odML framework with another DAQ system, such as Cerebus from Blackrock Microsystems, naturally comprised the work to implement a corresponding odML-file structure and adapt the compilation procedures to create a comprehensive metadata collection. In addition, manual metadata entries and time-delayed additions to the metadata collection were indeed possible, but not originally anticipated in the usage of the odML framework, and the corresponding workflow is therefore cumbersome. The need to develop odMLtables for the R2G experiment shows that the collection of metadata in standardized frameworks has to be easier and more intuitive across a larger variety of experiments, and is best accompanied by an up-to-date graphical user interface. To support the request to optimize the current metadata acquisition, and to provide a foundation of common knowledge about the experimental workflow and needs for metadata management of experimentalists in the field of neuroscience, and in particular electrophysiology, I suggested to set up a corresponding online survey as part of the Bachelor thesis of J. Pick. She recently developed the survey in collaboration with Dr. M. Denker, J. Sprenger, C. Canova, and me, and was able to collect

responses from 20 researchers within the time from August 4th to August 11th, 2016. As of today (Wednesday 30$^{\text{th}}$ November, 2016), the survey is still available online[16]. We plan to collect further responses to analyse the derived user requirements for metadata acquisition system to increase the awareness of software engineers for the need to optimize available metadata frameworks for neuroscientific experiments and provide them with more insight into the perspective of the actual users of the software, namely experimentalists. Based on these responses from August 11th, J. Pick derived a first set of user requirements for metadata acquisition system of which I picked the following examples to support my argument, that one has to facilitate collecting metadata in established frameworks (for the corresponding questions and responses see Figure 5.2):

- From the responses to Question 29 "*When do you manually register metadata?*", one can derive, first, from the fact that all 20 participants answered that metadata are actually manually registered, and, second, from the fact that more than half of the participants selected at least 2 answers that this manual metadata registration happens at different occasions in the course of the experiment. Thus, an important feature for metadata frameworks is indeed to allow the user to manually enter metadata, and be flexible enough to handle manually entries at different points in time (cf. reqt 5 in Section 1.2, and case 1 in Section 3.1).
- From the responses to Question 42 "*Which formats are used for electronically available metadata?*", one can derive, that indeed spreadsheet formats (13 responses) and text formats (11 responses) are the most commonly used metadata formats. The question remains if a metadata framework could predetermine the structure of the metadata in these formats to be able to provide generic conversion routines as, e.g. introduced in odMLtables for csv and xls (cf. Section 3.2.3). Currently available metadata frameworks are not used, which demonstrates that either these tools are not known, or that their usability is not yet suitable for integrating the tools in the daily lab routines.
- From the responses to Question 53 "*If you digitally register and represent metadata using a particular software tool, how important to you is the feature to print metadata or export them as pdf-file?*" (adopted formulation), one can derive from the fact that 11 participants answered with "*necessary*" or "*important*" and further 6 answered with "*nice to have*", that the possibility to create printable metadata overviews should be provided by a metadata framework.

Based on only these three selected user requirements of the up to date online survey, the acquisition of metadata into a comprehensive collection needs to be improved, if metadata frameworks want to obtain more acceptance in the neuroscientific community. Providing a user friendly solution for manually registering metadata, ideally based on commonly used formats (e.g., spreadsheets), should be a key issue in the development of new metadata frameworks. The other important component of an improved metadata framework, which can be extracted from the selected user requirements and will be further discussed in the next paragraph, is the possibility to visualize metadata.

Facilitation of metadata screening and interactive, metadata-based data selection In Case 3 in Section 3.1, I argue that gaining overviews of metadata of an entire experiment or also only of one dataset can be an important step in understanding the potential of the experimental data for research analyses. The survey conducted by J. Pick indeed supports this argument. For the question "*If you digitally register and represent metadata using a particular software tool, how important for you is the feature to generate plots or graphics?*" (adopted formulation), 6 participants

[16] https://goo.gl/forms/vU8r8AF75orEoB923

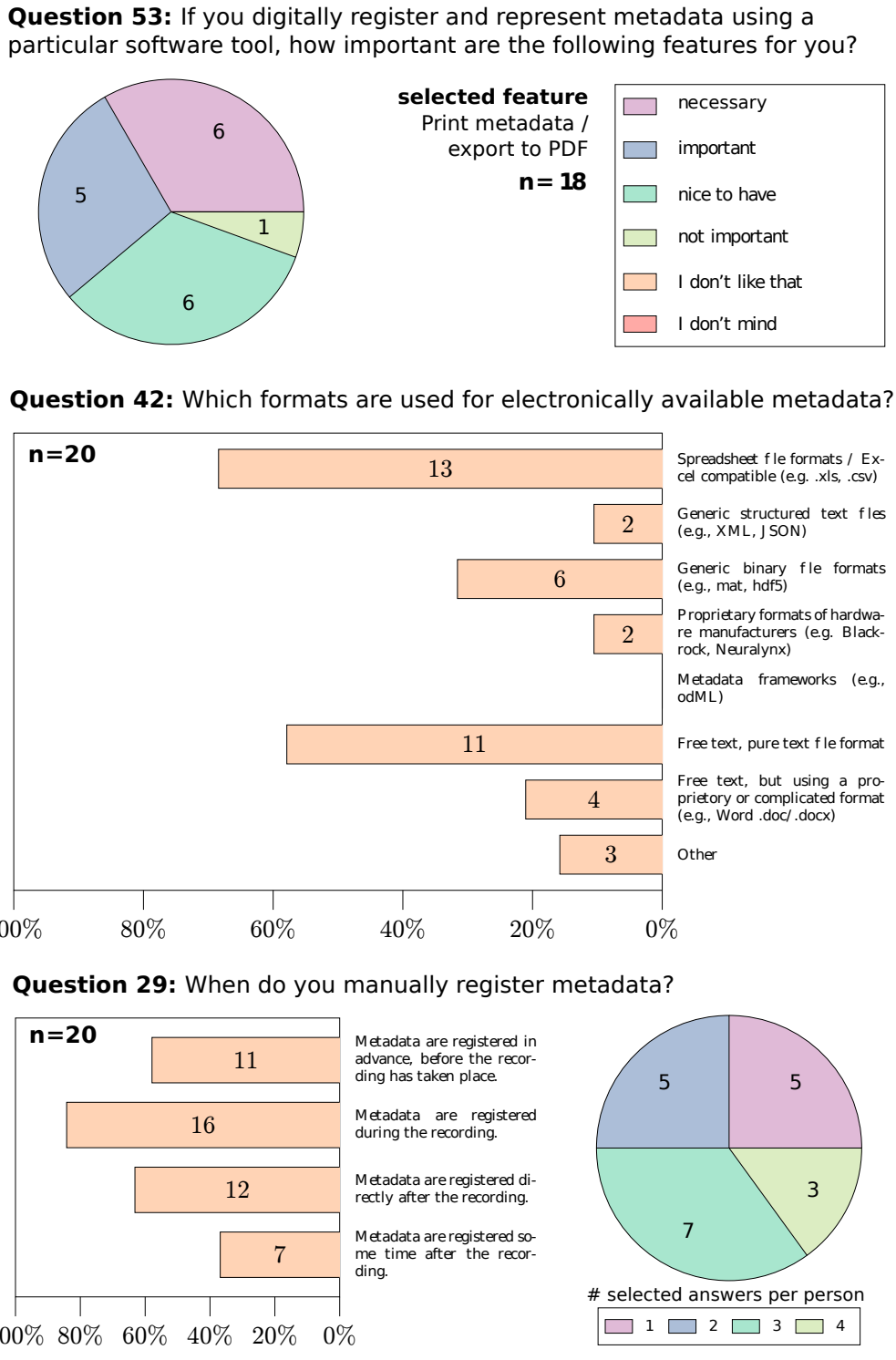

Figure 5.2 – Survey on metadata acquisition and handling in neuroscience - selected questions. (created from figures 4.8, 4.9, and 4.29 of the Bachelor thesis by J. Pick) The figure displays three selected questions and corresponding responses, which have been received between August 4th until August 11th. Among the 20 participants of the survey, were no undergraduates, 11 Ph.D. students, 2 post-docs, 4 staff scientists, 2 group leaders, and 1 professor. The age of most participants ranged between 26 and 45. Most participants (17) have an educational background in biology.

answered "*necessary*" or "*important*", and 9 participant at least agree that this feature is "*nice to have*". Furthermore, for the question "*If you digitally register and represent metadata using a particular software tool, how important for you is the feature to be able to perform calculations?*" (adopted formulation), 7 participants answered with "*necessary*" or "*important*", and 8 with "*nice to have*". The visualization of metadata overviews, including secondary metadata from small calculations (e.g., the reaction time as difference between time stamps of two trigger events), in either a tabular (cf. Figure 3.7, and Question 53 in Figure 5.2) or a graphical (cf. Figure 3.3) format is currently not directly supported by the odML framework, and is also in other software solutions for metadata management not included. The `iter` and `filter`-functions of the odML framework (cf. Section 3.3.2) at least provide a proper filter and navigation of the corresponding metadata structures, which can be used to visualize metadata in a secondary process. Nevertheless, integrating corresponding interactive functionalities directly into metadata frameworks would facilitate metadata screening procedures and with this, would increase distinctly the usability of the management tools for the researchers. Moreover, such visualization features would have an even greater impact, if they would provide the possibility to perform and illustrate metadata based data selection as argued in Case 4 of Section 3.1. For this, a corresponding software would need to require the functional connection between the used metadata and data framework, which is easiest to establish when the both frameworks are integrated into one as realized in the NIX framework. In fact, developers of the NIX framework added a first attempt to provide metadata guided visualization of data in form of a corresponding generic viewer, called NixView[17]. Another software example, where the user has the possibility to gain immediate insights from comparing analysis results of multiple visualizations created for different data selections, is VisTrails[18].

Better bindings between data, metadata, and workflow management solutions In Section 1.3, I discuss that for neuroscientific experiments the zoo of file formats makes it particularly difficult to set up generic and robust data loading routines (Garcia et al., 2014; Denker and Grün, 2015). The Neo framework provides a solution by loading data into an internal, uniform structure which can be used to access and analyse data independently of their original format. In Section 2.5.1, I demonstrate for the example experiment that metadata storage results in a similar, if not even more heterogeneous, zoo of file formats, if not differently managed via a standardized framework, such as odML. However, for improving an actual analysis workflow one has to first merge the used data and metadata frameworks in a meaningful way. Puzzling out how to implement the data and metadata frameworks and in addition, how connect them, is currently a task scientists have to manage for themselves. Furthermore, other tools, such as common analysis toolboxes (e.g., Elephant) and provenance trail software (e.g., Sumatra), require the same effort when they are integrated into a scientific workflow, because they need not only to capture the semantic context of the research analysis, but also need to be robustly associated with the correct data via a correspondingly managed metadata system (e.g., Koop et al., 2010). In order that all these techniques for data, metadata, and workflow management, of which I only introduce a few in my thesis, are actually accepted and used by the neuroscientific community, an important step would be to provide a decent infrastructure for these software solutions. Such an infrastructure has to propagate what solutions are available, how one can implement them, but most importantly, has to demonstrate how the different pieces of this software puzzle can be linked.

[17] https://github.com/bendalab/nixview

[18] https://www.vistrails.org/index.php/Main_Page

In my opinion successful policies on how to accomplish reproducible neuroscience will highly depend on the community effort to find working solutions. In this respect, I believe that the "community" has to consist of experimentalists, as well as software engineers, and computational and/or theoretical neuroscientists. With my thesis, I provide a concrete use case for such a development. Nonetheless, I think, that, in addition, funding agencies and institutions have to not only request proper data, metadata, and workflow management, but have to actually help researchers to practise them. A corresponding support could be provided, for example by providing training in up-to-date technologies and explicit founding of additional time and/or manpower to implement the available software solutions. With a proper support and a better knowledge of how to reproducibly manage data, metadata and performed analysis workflows, neuroscientists will grow in confidence to be more willing to openly share their data with third parties. Due to the increasing complexity of experiments in the highly interdisciplinary field of neuroscience, I make the assertion that data sharing will be a necessity in future to establish complete open scientific inquiries and further promote scientific progress.

Acknowledgements

Financially, this work was partly supported by the Helmholtz Portfolio "Supercomputing and Modeling for the Human Brain" (SMHB), the Human Brain Project (HBP, EU Grant 604102), the German Neuroinformatics Node (G-Node, BMBF Grant 01GQ1302), BrainScaleS (EU Grant 269912), the DFG SPP Priority Program 1665 (GR 1753/4-1 and DE 2175/1-1), the Collaborative Research Agreement RIKEN-CNRS, ANR GRASP, CNRS (PEPS, Neuro_IC2010) and DAAD.

Personally, I would like to thank Prof. Dr. Sonja Grün for her support in supervising my thesis, and for giving me, as a biological neuroscientist, the chance to confer a doctorate in such an interdisciplinary topic. I would also like to thank her for providing a work environment which allowed me to expand my connections to a variety of collaboration partners in the field of neuroscience and intensified my experience in interdisciplinary communication.

My deepest gratitude extends to Dr. Michael Denker for all the countless hours of discussions and enormous amount of work he has invested towards my training in computational and statistical aspects of neuroscience. I profited from his patience and irrepressible optimism!

I am also deeply grateful for the close collaboration with Dr. Thomas Brochier. He always supported and endured my interrogation methods while I was reshaping the data and metadata management of his research project and supervised the biological aspects of my thesis.

Furthermore, I am in debt to Dr. Benjamin Weyers for supporting me to grasp all the computation aspects of my project, for donating his time for inspiring discussions with me about my thesis, and for always pushing me to believe in my abilities!

My sincere thanks also goes to Prof. Dr. Abigail Morrison for mentally supporting me during the last months and revising my thesis.

I have greatly benefited from supervising Jana Pick during her Bachelor studies. She was a great support in shaping odMLtables towards a promising open-source project.

In this respect, I would also like to thank Julia Sprenger who I enjoyed and enjoy working with not only for optimizing and extending the functionalities of odMLtables, but also for supporting the implementations of the `BlackrockIO` and `ReachGraspIO`.

I thank my fellow lab mates at the INM-6/IAS-6 for all the stimulating and motivational discussions on science and other topics, and for all the fun we have had in the last four years!

My special thanks goes also to the members of the CoMCo group in Marseille and the members of the G-Node in Munich. They all were extraordinarily tolerant and supportive of my work, provided me with invaluable feedback, and always welcomed me in their laboratories.

In general I am thankful to all my colleagues and friends inside and outside of the scientific world for their constant support and encouragement, especially towards the end of my thesis. In particular, I am grateful for my long-time office mate Dr. Emiliano Torre for not only being an inspiring colleague with whom I could always share my ideas and on who's feedback I could always rely on, but also for being a close friend from the very beginning of the doctoral studies.

I commemorate Dr. Paul Chorley and thank him for the countless inspiring discussions over coffee breaks, his irrepressible and infectious enthusiasm, and for his valuable input during the development of the metadata structures and templates of the odML metadata collection.

Last but not least, my deepest heartfelt appreciation goes to my family: my mother, my brother and my sister and her family for supporting me mentally throughout writing this thesis and my life in general. I dedicate my thesis to them and in memory to my father with whom I had countless guiding discussions during my education that shaped me into the person I am today.

Bibliography

Adams, J. (2012). Collaborations: The rise of research networks. *Nature*, 490(7420):335–336.

Alsheikh-Ali, A. A., Qureshi, W., Al-Mallah, M. H., and Ioannidis, J. P. A. (2011). Public Availability of Published Research Data in High-Impact Journals. *PLoS ONE*, 6(9):e24357.

Aoki-Kinoshita, K. F., Kinjo, A. R., Morita, M., Igarashi, Y., Chen, Y.-a., Shigemoto, Y., Fujisawa, T., Akune, Y., Katoda, T., Kokubu, A., Mori, T., Nakao, M., Kawashima, S., Okamoto, S., Katayama, T., and Ogishima, S. (2015). Implementation of linked data in the life sciences at BioHackathon 2011. *Journal of Biomedical Semantics*, 6(1):3.

Baca, M. (2008). *Introduction to Metadata*, volume 3.0. Getty Publications, online edition edition.

Badia, R., Davison, A., Denker, M., Giesler, A., Gosh, S., Goble, C., Grewe, J., Grün, S., Hatsopoulos, N., LeFranc, Y., Muller, J., Pröpper, R., Teeters, J., Wachtler, T., Weeks, M., and Zehl, L. (2015). INCF Program on Standards for data sharing: New perspectives on workflows and data management for the analysis of electrophysiological data. Technical report, International Neuroinformatics Coordination Facility (INCF).

Baker, M. (2015). Reproducibility crisis: Blame it on the antibodies. *Nature*, 521(7552):274–276.

Bavoil, L., Callahan, S., Crossno, P., Freire, J., Scheidegger, C., Silva, C., and Vo, H. (2005). VisTrails: Enabling Interactive Multiple-View Visualizations. pages 135–142. IEEE.

Berenyi, A., Somogyvari, Z., Nagy, A. J., Roux, L., Long, J. D., Fujisawa, S., Stark, E., Leonardo, A., Harris, T. D., and Buzsaki, G. (2013). Large-scale, high-density (up to 512 channels) recording of local circuits in behaving animals. *Journal of Neurophysiology*, 111(5):1132–1149.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked Data - The Story So Far:. *International Journal on Semantic Web and Information Systems*, 5(3):1–22.

Bloom, T., Ganley, E., and Winker, M. (2014). Data Access for the Open Access Literature: PLOS's Data Policy. *PLoS Biology*, 12(2):e1001797.

Borgman, C. L. (2010). Research Data: Who Will Share What, with Whom, When, and Why? *SSRN Electronic Journal*.

Borgman, C. L. (2012). The conundrum of sharing research data. *Journal of the American Society for Information Science and Technology*, 63(6):1059–1078.

Bowden, D. M., Dubach, M., and Park, J. (2007). Creating Neuroscience Ontologies. In Walker, J. M., editor, *Neuroinformatics*, volume 401, pages 67–87. Humana Press, Totowa, NJ.

Brammer, G. R., Crosby, R. W., Matthews, S. J., and Williams, T. L. (2011). Paper Mâché: Creating Dynamic Reproducible Science. *Procedia Computer Science*, 4:658–667.

Brochier, T., Zehl, L., Hao, Y., Duret, M., Sprenger, J., Denker, M., Grün, S., and Riehle, A. (in prep). Massively parallel multi-electrode recordings of macaque motor cortex during an instructed delayed reach-to-grasp task.

Buzsáki, G. and Draguhn, A. (2004). Neuronal Oscillations in Cortical Networks. *Science*, 304(5679):1926–1929.

Campbell, E. G., Clarridge, B. R., Gokhale, M., Birenbaum, L., Hilgartner, S., Holtzman, N. A., and Blumenthal, D. (2002). Data Withholding in Academic Genetics: Evidence From a National Survey. *JAMA*, 287(4):473.

Candela, L., Castelli, D., Manghi, P., and Tani, A. (2015). Data journals: A survey. *Journal of the Association for Information Science and Technology*, 66(9):1747–1762.

Chacon, S. and Straub, B. (2014). *Pro Git*. Apress, 2nd ed. edition edition.

Cowley, B., Kaufman, M., Butler, Z., Churchland, M., Ryu, S., Shenoy, K., and Yu, B. (2013). DataHigh: Graphical user interface for visualizing and interacting with high-dimensional neural activity. *Journal of Neural Engineering*.

Crook, S. M., Bednar, J. A., Berger, S., Cannon, R., Davison, A. P., Djurfeldt, M., Eppler, J., Kriener, B., Furber, S., Graham, B., Plesser, H. E., Schwabe, L., Smith, L., Steuber, V., and van Albada, S. (2012). Creating, documenting and sharing network models. *Network*, 23(4):131–149.

Curcin, V. and Ghanem, M. (2008). Scientific workflow systems - can one size fit all? pages 1–9. IEEE.

Dallmeier-Tiessen, S., Darby, R., Gitmans, K., Lambert, S., Suhonen, J., and Wilson, M. (2012). Compilation of results on drivers and barriers and new opportunities. http://www.alliancepermanentaccess.org/wp-content/uploads/downloads/2012/08/ODE-CompilationResultsDriversBarriersNewOpportunities1.pdf.

Davidson, S. B. and Freire, J. (2008). Provenance and scientific workflows: Challenges and opportunities. page 1345. ACM Press.

Davison, A. (2012). Automated Capture of Experiment Context for Easier Reproducibility in Computational Research. *Computing in Science & Engineering*, 14(4):48–56.

Deisseroth, K. and Schnitzer, M. J. (2013). Engineering approaches to illuminating brain structure and dynamics. *Neuron*, 80(3):568–577.

Denker, M., Einevoll, E., Franke, F., Grün, S., Hagen, E., Kerr, J., Nawrot, M., Ness, T., Ritz, R., Smith, L., Wachtler, T., and Wojcik, D. (2014a). 1st INCF workshop on validation of analysis methods. Technical report, International Neuroinformatics Coordination Facility (INCF).

Denker, M. and Grün, S. (2015). Designing workflows for the reproducible analysis of electrophysiological data. In *Pro- Ceedings of the International Workshop of Brain-Inspired Computing 2015*, Lecture Notes in Computer Science, page (in press). Springer.

Denker, M., Riehle, A., Diesmann, M., and Grün, S. (2010). Estimating the contribution of assembly activity to cortical dynamics from spike and population measures. *Journal of Computational Neuroscience*, 29(3):599–613.

Denker, M., Roux, S., Linden, H., Diesmann, M., Riehle, A., and Grun, S. (2011). The Local Field Potential Reflects Surplus Spike Synchrony. *Cerebral Cortex*, 21(12):2681–2695.

Denker, M., Roux, S., Timme, M., Riehle, A., and Grün, S. (2007). Phase Synchronization between LFP and Spiking Activity in Motor Cortex during Movement Preparation. *Neurocomputing*, 70(10-12):2096–2101.

Denker, M., Yegenoglu, A., Holstein, D., Torre, E., Jennings, T., Davison, A., and Grün, S. (2015a). Elephant: An open-source tool for the analysis of electrophysiological data.}. In *Proceedings of the 11th Meeting of the German Neuroscience Society, Neuroforum 2015*, pages T27–2B.

Denker, M., Zehl, L., Kilavik, B., Diesmann, M., Brochier, T., Riehle, A., and Grün, S. (2014b). Characterizing spatially organized LFP beta oscillations in the macaque motor cortex. In *AREADNE Conference 2014, Santorini, Greece, 25-29 June 2014*, page 62, Cambridge, Massachusetts, USA. The AREADNE Foundation, Inc.

Denker, M., Zehl, L., Kilavik, B., Diesmann, M., Brochier, T., Riehle, A., and Grün, S. (2015b). Relating spatial patterns of beta oscillations to their power in macaque motor cortex: A case study in using the "Elephant" data analysis framework in a reproducible analysis workflow. In *SfN 2015, Monday, Scientific Session Listings 269-451*, volume Program No. 293.06, page Poster No. A95, Chicago, IL. online. Society for Neuroscience 45th Annual Meeting, Chicago, IL.

Denker, M., Zehl, L., Kilavik, B. E., Diesmann, M., Brochier, T., Riehle, A., and Grün, S. (in prep). The amplitude of beta spindles determines the mesoscopic spatial organization of beta oscillations - Spindles tell about spatial organization of beta oscillations.

Doldirina, C., Ostlaender, N., Perego, A., Annoni, A., Kanellopoulos, I., Craglia, M., Friis-Christensen, A., Vaccari, L., Tartaglia, G., Bonato, F., Triaille, J.-P., and Gentile, S. (2015). JRC Data Policy. *Publications Office of the European Union*, 2015(9):1–14.

Dotson, N. M., Salazar, R. F., and Gray, C. M. (2014). Frontoparietal Correlation Dynamics Reveal Interplay between Integration and Segregation during Visual Working Memory. *Journal of Neuroscience*, 34(41):13600–13613.

Durka, P. J. and Ircha, D. (2004). SignalML: Metaformat for description of biomedical time series. *Computer Methods and Programs in Biomedicine*, 76(3):253–259.

Editorial (2005). The cost of salami slicing. *Nature Materials*, 4(1):1–1.

Einevoll, G. T., Kayser, C., Logothetis, N. K., and Panzeri, S. (2013). Modelling and Analysis of Local Field Potentials for Studying the Function of Cortical Circuits. *Nature Reviews Neuroscience*, 14(11):770–785.

Engel, A. K., König, P., Gray, C. M., and Singer, W. (1990). Stimulus-Dependent Neuronal Oscillations in Cat Visual Cortex: Inter-Columnar Interaction as Determined by Cross-Correlation Analysis. *Eur. J. Neurosci.*, 2(7):588–606.

Ermentrout, G. B. and Kleinfeld, D. (2001). Traveling electrical waves in cortex: Insights from phase dynamics and speculation on a computational role. *Neuron*, 29(1):33–44.

Fanelli, D. (2010). Do Pressures to Publish Increase Scientists' Bias? An Empirical Support from US States Data. *PLoS ONE*, 5(4):e10271.

Fienberg, S. E., Martin, M. E., Straf, M. L., National Research Council (U.S.), and Committee on National Statistics (1985). *Sharing Research Data*. National Academy Press, Washington, D.C.

Foundation, N. S. (2014). Dissemination and Sharing of Research Results | NSF - National Science Foundation. http://www.nsf.gov/bfa/dias/policy/dmp.jsp.

Freeman, W. J. (1978). Spatial Properties of an EEG Event in the Olfactory Bulb and Cortex. *Electroencephalogr Clin Neurophysiol*, 44(5):586–605.

Freitas, A., Curry, E., Oliveira, J. G., and O'Riain, S. (2012). Querying Heterogeneous Datasets on the Linked Data Web: Challenges, Approaches, and Trends. *IEEE Internet Computing*, 16(1):24–33.

Friedrich, R. W., Habermann, C. J., and Laurent, G. (2004). Multiplexing Using Synchrony in the Zebrafish Olfactory Bulb. *Nature Neuroscience*, 7(8):862–871.

Fries, P. (2005). A Mechanism for Cognitive Dynamics: Neuronal Communication through Neuronal Coherence. *Trends in Cognitive Sciences*, 9(10):474–480.

Fries, P. (2015). Rhythms for Cognition: Communication through Coherence. *Neuron*, 88(1):220–235.

Fries, P., Reynolds, J. H., Rorie, A. E., and Desimone, R. (2001). Modulation of oscillatory neuronal synchronization by selective visual attention. *Science*, 291(5508):1560–1563.

Friston, K. J., Bastos, A. M., Pinotsis, D., and Litvak, V. (2015). LFP and Oscillations—what Do They Tell Us? *Current Opinion in Neurobiology*, 31:1–6.

Garcia, S. (2009). OpenElectrophy: An electrophysiological data and analysis sharing framework. *Frontiers in Neuroinformatics*, 3.

Garcia, S., Guarino, D., Jaillet, F., Jennings, T., Pröpper, R., Rautenberg, P. L., Rodgers, C. C., Sobolev, A., Wachtler, T., Yger, P., and Davison, A. P. (2014). Neo: An object model for handling electrophysiology data in multiple formats. *Frontiers in Neuroinformatics*, 8.

Geisler, W. S. (2008). Visual Perception and the Statistical Properties of Natural Scenes. *Annu. Rev. Psychol.*, 59(1):167–192.

Gil, Y., Ratnakar, V., Kim, J., Gonzalez-Calero, P., Groth, P., Moody, J., and Deelman, E. (2011). Wings: Intelligent Workflow-Based Design of Computational Experiments. *IEEE Intelligent Systems*, 26(1):62–72.

Glasson, E. J. and Hussain, R. (2008). Linked data: Opportunities and challenges in disability research. *Journal of Intellectual and Developmental Disability*, 33(4):285–291.

Gleeson, P., Crook, S., Cannon, R. C., Hines, M. L., Billings, G. O., Farinella, M., Morse, T. M., Davison, A. P., Ray, S., Bhalla, U. S., Barnes, S. R., Dimitrova, Y. D., and Silver, R. A. (2010). NeuroML: A language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS Comput Biol*, 6(6):e1000815.

Greene, M. (2007). The demise of the lone author. *Nature*, 450(7173):1165–1165.

Grewe, J., Wachtler, T., and Benda, J. (2011). A Bottom-up Approach to Data Annotation in Neurophysiology. *Frontiers in Neuroinformatics*, 5(16).

Grün, S. (2009). Data-driven significance estimation of precise spike correlation. *JNeurophysiol*, (101):1126–1140. (invited review).

Guarino, N., Oberle, D., and Staab, S. (2009). What Is an Ontology? In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg.

Halb, W., Raimond, Y., and Hausenblas, M. (2008). Building Linked Data For Both Humans and Machines. http://data.semanticweb.org/workshop/LDOW/2008/paper/10.

Hanson, B., Sugden, A., and Alberts, B. (2011). Making Data Maximally Available. *Science*, 331(6018):649–649.

Hatsopoulos, N., Joshi, J., and O'Leary, J. G. (2004). Decoding Continuous and Discrete Motor Behaviors Using Motor and Premotor Cortical Ensembles. *Journal of Neurophysiology*, 92(2):1165–1174.

Heitmann, S., Gong, P., and Breakspear, M. (2012). A Computational Role for Bistability and Traveling Waves in Motor Cortex. *Frontiers in Computational Neuroscience*, 6:67.

Hines, W. C., Su, Y., Kuhn, I., Polyak, K., and Bissell, M. J. (2014). Sorting Out the FACS: A Devil in the Details. *Cell Reports*, 6(5):779–781.

Hosaka, R., Nakajima, T., Aihara, K., Yamaguchi, Y., and Mushiake, H. (2015). The Suppression of Beta Oscillations in the Primate Supplementary Motor Complex Reflects a Volatile State During the Updating of Action Sequences. *Cerebral Cortex*, page bhv163.

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., and others (2003). The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531.

Ito, J. (2014). Spike-Triggered Average. In Jaeger, D. and Jung, R., editors, *Encyclopedia of Computational Neuroscience*, pages 1–5. Springer New York, New York, NY.

Kemp, B. and Olivan, J. (2003). European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data. *Clinical Neurophysiology*, 114(9):1755–1761.

Kemp, B., Värri, A., Rosa, A. C., Nielsen, K. D., and Gade, J. (1992). A simple format for exchange of digitized polygraphic recordings. *Electroencephalography and Clinical Neurophysiology*, 82(5):391–393.

Kilavik, B. E., Ponce-Alvarez, A., Trachel, R., Confais, J., Takerkart, S., and Riehle, A. (2012). Context-Related Frequency Modulations of Macaque Motor Cortical LFP Beta Oscillations. *Cerebral Cortex*, 22(9):2148–2159.

Kilavik, B. E., Zaepffel, M., Brovelli, A., MacKay, W. A., and Riehle, A. (2013). The ups and downs of beta oscillations in sensorimotor cortex. *Experimental Neurology*, 245:15–26.

Kilner, J., Bott, L., and Posada, A. (2005). Modulations in the degree of synchronization during ongoing oscillatory activity in the human brain. *European Journal of Neuroscience*, 21(9):2547–2554.

Kim, U., Bal, T., and McCormick, D. A. (1995). Spindle Waves Are Propagating Synchronized Oscillations in the Ferret LGNd in Vitro. *Journal of Neurophysiology*, 74(3):1301–1323.

Klump, J. (2012). Offener Zugang zu Forschungsdaten. In Herb, U., editor, *Open Initiatives*, pages 45–53. universaar.

Koop, D., Santos, E., Bauer, B., Troyer, M., Freire, J., and Silva, C. T. (2010). Bridging Workflow and Data Provenance Using Strong Links. In Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Gertz, M., and Ludäscher, B., editors, *Scientific and Statistical Database Management*, volume 6187, pages 397–415. Springer Berlin Heidelberg, Berlin, Heidelberg.

Koop, D., Santos, E., Mates, P., Vo, H. T., Bonnet, P., Bauer, B., Surer, B., Troyer, M., Williams, D. N., Tohline, J. E., Freire, J., and Silva, C. T. (2011). A Provenance-Based Infrastructure to Support the Life Cycle of Executable Papers. *Procedia Computer Science*, 4:648–657.

Köster, J. and Rahmann, S. (2012). Snakemake–a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522.

Kuipers, T. and van der Hoeven, J. (2009). PARSE.Insight. Deliverable D3.4 Survey Report. of research output Europe Title of Deliverable Survey Report. http://www.yumpu.com/en/document/view/10441813/parseinsight-survey-report.

Kwan, H., Mackay, W., Murphy, I., and Wong, Y. (1978). An intracortical microstimulation study of output organization in precentral cortex of awake primates. *Journal de physiologie*, 74(3):231–233.

Laine, C. (2007). Reproducible Research: Moving toward Research the Public Can Really Trust. *Ann Intern Med*, 146(6):450.

Larson, S. D. (2009). Ontologies for neuroscience: What are they and what are they good for? *Frontiers in Neuroscience*, 3(1).

Le Franc, Y., Gonzalez, D., Mylyanyk, I., Grewe, J., Jezek, P., Moŭcek, R., and Wachtler, T. (2014). Mobile metadata: Bringing Neuroinformatics tools to the bench. *Frontiers in Neuroinformatics*, 8.

Lewis, C., Bosman, C., and Fries, P. (2015). Recording of brain activity across spatial scales. *Current Opinion in Neurobiology*, 32:68–77.

Leydesdorff, L. and Wagner, C. S. (2008). International collaboration in science and the formation of a core group. *Journal of Informetrics*, 2(4):317–325.

Lisman, J. (2015). The Challenge of Understanding the Brain: Where We Stand in 2015. *Neuron*, 86(4):864–882.

Logothetis, N. K. and Wandell, B. A. (2004). Interpreting the BOLD Signal. *Annual Review of Physiology*, 66(1):735–769.

Louis, S., Gerstein, G. L., Grün, S., and Diesmann, M. (2010). Surrogate spike train generation through dithering in operational time. *Frontiers in Compututational Neuroscience*, 4(127).

Maldonado, P., Babul, C., Singer, W., Rodriguez, E., Berger, D., and Grün, S. (2008). Synchronization of Neuronal Responses in Primary Visual Cortex of Monkeys Viewing Natural Images. *Journal of Neurophysiology*, 100(3):1523–1532.

Masquelier, T., Hugues, E., Deco, G., and Thorpe, S. J. (2009). Oscillations, Phase-of-Firing Coding, and Spike Timing-Dependent Plasticity: An Efficient Learning Scheme. *Journal of Neuroscience*, 29(43):13484–13493.

Milekovic, T., Truccolo, W., Grün, S., Riehle, A., and Brochier, T. (2015). Local field potentials in primate motor cortex encode grasp kinetic parameters. *NeuroImage*, 114:338–355.

Mitzdorf, U. (1985). Current source-density method and application in cat cerebral cortex: Investigation of evoked potentials and EEG phenomena. *Physiological Review*.

Miyamoto, D. and Murayama, M. (2015). The fiber-optic imaging and manipulation of neural activity during animal behavior. *Neuroscience Research*.

Morrison, S. J. (2014). Time to do something about reproducibility. *eLife*, 3:e03981.

Muller, L., Reynaud, A., Chavane, F., and Destexhe, A. (2014). The stimulus-evoked population response in visual cortex of awake monkey is a propagating wave. *Nature Communications*, 5.

Murthy, V. N. and Fetz, E. E. (1992). Coherent 25- to 35-Hz Oscillations in the Sensorimotor Cortex of Awake Behaving Monkeys. *Proceedings of the National Academy of Sciences*, 89(12):5670–5674.

Murthy, V. N. and Fetz, E. E. (1996a). Oscillatory Activity in Sensorimotor Cortex of Awake Monkeys: Synchronization of Local Field Potentials and Relation to Behavior. *Journal of Neurophysiology*, 76(6):3949–3967.

Murthy, V. N. and Fetz, E. E. (1996b). Synchronization of Neurons during Local Field Potential Oscillations in Sensorimotor Cortex of Awake Monkeys. *Journal of Neurophysiology*, 76(6):3968–3982.

Nakajima, T., Hosaka, R., Tsuda, I., Tanji, J., and Mushiake, H. (2013). Two-Dimensional Representation of Action and Arm-Use Sequences in the Presupplementary and Supplementary Motor Areas. *Journal of Neuroscience*, 33(39):15533–15544.

National Institutes of Health (2003). NIH Data Sharing Policy and Implementation Guidance. https://grants.nih.gov/grants/policy/data_sharing/data_sharing_guidance.htm.

Nature (2005). Let data speak to data. *Nature*, 438(7068):531–531.

Nauhaus, I., Busse, L., Carandini, M., and Ringach, D. L. (2009). Stimulus contrast modulates functional connectivity in visual cortex. *Nature Neuroscience*, 12(1):70–76.

Neill, U. S. (2008). Publish or perish, but at what cost? *Journal of Clinical Investigation*, 118(7):2368–2368.

Nicolelis, M. A. L. and Ribeiro, S. (2002). Multielectrode recordings: The next steps. *Curr Opin Neurobiol*, 12(5):602–606.

Nordhausen, C. T., Maynard, E. M., and Normann, R. A. (1996). Single unit recording capabilities of a 100 microelectrode array. *Brain Research*, 726(1–2):129–140.

Nowakowski, P., Ciepiela, E., Harężlak, D., Kocot, J., Kasztelnik, M., Bartyński, T., Meizner, J., Dyk, G., and Malawski, M. (2011). The Collage Authoring Environment. *Procedia Computer Science*, 4:608–617.

Obien, M. E. J., Deligkaris, K., Bullmann, T., Bakkum, D. J., and Frey, U. (2014). Revealing neuronal function through microelectrode array recordings. *Front Neurosci*, 8:423.

Open Science Collaboration (2015). Estimating the reproducibility of psychological science. *Science*, 349(6251):aac4716–aac4716.

Pampel, H. and Dallmeier-Tiessen, S. (2014). Open Research Data: From Vision to Practice. In Bartling, S. and Friesike, S., editors, *Opening Science*, pages 213–224. Springer International Publishing, Cham.

Park, M., Belhaj-Saïf, A., Gordon, M., and Cheney, P. (2001). Consistent features in the forelimb representation of primary motor cortex in rhesus macaques. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 21(8):2784–2792.

Paton, N. W. (2008). Managing and sharing experimental data: Standards, tools and pitfalls. *Biochemical Society Transactions*, 36(1):33–36.

Peng, R. D. (2011). Reproducible Research in Computational Science. *Science*, 334(6060):1226–1227.

Pesaran, B., Pezaris, J. S., Sahani, M., Mitra, P. P., and Andersen, R. A. (2002). Temporal Structure in Neuronal Activity during Working Memory in Macaque Parietal Cortex. *Nature Neuroscience*, 5(8):805–811.

Pfurtscheller, G. and Aranibar, A. (1979). Evaluation of event-related desynchronization (ERD) preceding and following voluntary self-paced movement. *Electroencephalography and Clinical Neurophysiology*, 46(2):138–146.

Poldrack, R. A. and Gorgolewski, K. J. (2014). Making big data open: Data sharing in neuroimaging. *Nature Neuroscience*, 17(11):1510–1517.

Pröpper, R. and Obermayer, K. (2013). Spyke Viewer: A flexible and extensible platform for electrophysiological data analysis. *Frontiers in Neuroinformatics*, 7.

Pulverer, B. (2014). Transparent, Reproducible Data. *The EMBO Journal*, 33(22):2597–2597.

Pulverer, B. (2015a). Reproducibility blues. *The EMBO Journal*, 34(22):2721–2724.

Pulverer, B. (2015b). When things go wrong: Correcting the scientific record. *The EMBO Journal*, 34(20):2483–2485.

Riehle, A. (1997). Spike Synchronization and Rate Modulation Differentially Involved in Motor Cortical Function. *Science*, 278(5345):1950–1953.

Riehle, A., Wirtssohn, S., Grün, S., and Brochier, T. (2013). Mapping the spatio-temporal structure of motor cortical LFP and spiking activities during reach-to-grasp movements. *Frontiers in Neural Circuits*, 7.

Rougeul, A., Bouyer, J., Dedet, L., and Debray, O. (1979). Fast somato-parietal rhythms during combined focal attention and immobility in baboon and squirrel monkey. *Electroencephalography and Clinical Neurophysiology*, 46(3):310–319.

Rubino, D., Robbins, K. A., and Hatsopoulos, N. G. (2006). Propagating waves mediate information transfer in the motor cortex. *Nature Neuroscience*, 9(12):1549–1557.

Sato, T. K., Nauhaus, I., and Carandini, M. (2012). Traveling Waves in Visual Cortex. *Neuron*, 75(2):218–229.

Savage, C. J. and Vickers, A. J. (2009). Empirical Study of Data Sharing by Authors Publishing in PLoS Journals. *PLoS ONE*, 4(9):e7078.

Schwarz, D. A., Lebedev, M. A., Hanson, T. L., Dimitrov, D. F., Lehew, G., Meloy, J., Rajangam, S., Subramanian, V., Ifft, P. J., Li, Z., and al, e. (2014). Chronic, wireless recordings of large-scale brain activity in freely moving rhesus monkeys. *Nat Meth*, 11(6):670–676.

Singer, W. (1999). Neuronal Synchrony: A Versatile Code for the Definition of Relations? *Neuron*, 24(1):49–65.

Sobolev, A., Stoewer, A., Leonhardt, A., Rautenberg, P. L., Kellner, C. J., Garbers, C., and Wachtler, T. (2014a). Integrated platform and API for electrophysiological data. *Front Neuroinform*, 8:32.

Sobolev, A., Stoewer, A., Pereira, M., Kellner, C. J., Garbers, C., Rautenberg, P. L., and Wachtler, T. (2014b). Data management routines for reproducible research using the G-Node Python Client library. *Front Neuroinform*, 8:15.

Sprenger, J., Zehl, L., Canova, C., Pick, J., Bitzenhofer, S., Ahlbeck, J., Takagaki, K., Hanganu-Opatz, I., Ohl, F., Grün, S., and Denker, M. (in prep). odMLtables: A user-friendly approach to managing metadata of neurophysiological experiments.

Stodden, V., Leisch, F., and Peng, R. D., editors (2014). *Implementing Reproducible Research (Chapman & Hall/CRC The R Series)*. Chapman and Hall/CRC.

Stoewer, A., Kellner, C., Benda, J., Wachtler, T., and Grewe, J. (2014). File format and library for neuroscience data and metadata. *Frontiers in Neuroinformatics*, 8.

Suner, S., Fellows, M., Vargas-Irwin, C., Nakata, G., and Donoghue, J. (2005). Reliability of Signals From a Chronically Implanted, Silicon-Based Electrode Array in Non-Human Primate Primary Motor Cortex. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13(4):524–541.

Takahashi, K., Kim, S., Coleman, T. P., Brown, K. A., Suminski, A. J., Best, M. D., and Hatsopoulos, N. G. (2015). Large-Scale Spatiotemporal Spike Patterning Consistent with Wave Propagation in Motor Cortex. *Nature Communications*, 6:7169.

Teeters, J. L., Godfrey, K., Young, R., Dang, C., Friedsam, C., Wark, B., Asari, H., Peron, S., Li, N., Peyrache, A., Denisov, G., Siegle, J. H., Olsen, S. R., Martin, C., Chun, M., Tripathy, S., Blanche, T. J., Harris, K., Buzsáki, G., Koch, C., Meister, M., Svoboda, K., and Sommer, F. T. (2015). Neurodata Without Borders: Creating a Common Data Format for Neurophysiology. *Neuron*, 88(4):629–634.

Tejedor, E., Becerra, Y., Alomar, G., Queralt, A., Badia, R. M., Torres, J., Cortes, T., and Labarta, J. (2015). PyCOMPSs: Parallel computational workflows in Python. *International Journal of High Performance Computing Applications*.

Tenopir, C., Allard, S., Douglass, K., Aydinoglu, A. U., Wu, L., Read, E., Manoff, M., and Frame, M. (2011). Data Sharing by Scientists: Practices and Perceptions. *PLoS ONE*, 6(6):e21101.

Tomasello, M. and Call, J. (2011). Methodological Challenges in the Study of Primate Cognition. *Science*, 334(6060):1227–1228.

Torre, E., Picado-Muiño, D., Denker, M., Borgelt, C., and Grün, S. (2013). Statistical evaluation of synchronous spike patterns extracted by frequent item set mining. *Frontiers in Computational Neuroscience*, 7.

Torre, E., Quaglio, P., Denker, M., Brochier, T., Riehle, A., and Grun, S. (2016). Synchronous Spike Patterns in Macaque Motor Cortex during an Instructed-Delay Reach-to-Grasp Task. *Journal of Neuroscience*, 36(32):8329–8340.

Townsend, R. G., Solomon, S. S., Chen, S. C., Pietersen, A. N. J., Martin, P. R., Solomon, S. G., and Gong, P. (2015). Emergence of Complex Wave Patterns in Primate Cerebral Cortex. *Journal of Neuroscience*, 35(11):4657–4662.

Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods and applications.

Varela, F., Lachaux, J.-P., Rodriguez, E., and Martinerie, J. (2001). The Brainweb: Phase Synchronization and Large-Scale Integration. *Nature reviews neuroscience*, 2(4):229–239.

Vargas-Irwin, C. E., Shakhnarovich, G., Yadollahpour, P., Mislow, J. M. K., Black, M. J., and Donoghue, J. P. (2010). Decoding Complete Reach and Grasp Actions from Local Primary Motor Cortex Populations. *Journal of Neuroscience*, 30(29):9659–9669.

Verkhratsky, A., Krishtal, O. A., and Petersen, O. H. (2006). From Galvani to patch clamp: The development of electrophysiology. *Pflugers Arch*, 453(3):233–247.

Vidaurre, C., Sander, T. H., and Schlögl, A. (2011). BioSig: The Free and Open Source Software Library for Biomedical Signal Processing. *Computational Intelligence and Neuroscience*, 2011:1–12.

Webster, M. (2006). Merriam-Webster online dictionary.

Whitfield, J. (2008). Collaboration: Group theory. *Nature*, 455(7214):720–723.

Wicherts, J. M., Borsboom, D., Kats, J., and Molenaar, D. (2006). The poor availability of psychological research data for reanalysis. *American Psychologist*, 61(7):726–728.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M. P., Sufi, S., and Goble, C. (2013). The Taverna workflow suite: Designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(W1):W557–W561.

Womelsdorf, T., Schoffelen, J.-M., Oostenveld, R., Singer, W., Desimone, R., Engel, A. K., and Fries, P. (2007). Modulation of Neuronal Interactions Through Neuronal Synchronization. *Science*, 316(5831):1609–1612.

Zehl, L., Denker, M., Grün, S., Riehle, A., and Brochier, T. (2014a). Latency of LFP beta power peak during movement preparation correlates with reaction time. In *SfN 2014, Friday to Saturday, Scientific Session Listings 1-99*, volume Program No. 73.10, page Poster No. HH24, Washington, DC. online.

Zehl, L., Denker, M., Stoewer, A., Jaillet, F., Brochier, T., Riehle, A., Wachtler, T., and Grün, S. (2014b). Handling complex metadata in neurophysiological experiments. *Frontiers in Neuroinformatics*, 8.

Zehl, L., Jaillet, F., Stoewer, A., Grewe, J., Sobolev, A., Wachtler, T., Brochier, T. G., Riehle, A., Denker, M., and Grün, S. (2016). Handling Metadata in a Neurophysiology Laboratory. *Frontiers in Neuroinformatics*, 10:26.

Zhou, Y., De, S., Wang, W., and Moessner, K. (2016). Search Techniques for the Web of Things: A Taxonomy and Survey. *Sensors*, 16(5):600.

Appendix

Overview of R2G datasets that are going to be published

Within the interdisciplinary collaboration of the CoMCo group at the INT of the Aix-Marseille University in France and the SN group at the INM-6 of the Jülich Research Centre in Germany (for abbreviation, cf. beginning of Chapter 2), it was decided to publish two datasets of the R2G experiment in a data journal (e.g., Scientific Data): one dataset from monkey L (l101210-001) and one dataset from second recording period of monkey N (i140703-001).

					dur	
	date	**weekday**	**start**	**# rec**	rec day	rec*-001
monkey L	2010-12-10	Friday	10:50 am	9	01:28 h	11:49 min
monkey N	2014-07-03	Thursday	10:41 am	3	00:51 h	16:43 min

Table A.1 – Overview of recording days of the datasets that will be published. For both monkeys, we chose to publish the first dataset (rec*-001) of the recording day. For details on the published datasets see Table A.3 and Table A.2.

The dataset from monkey L, l101210-001, is the first out of 9 datasets recorded on Friday, December 10, 2010. The dataset from monkey N, i140703-001, is the first out of only 3 datasets recorded on Thursday, July 3, 2014. For the selected recording, both monkeys had to perform a reach-to-grasp task following the standard 2-inst paradigm with condition setting 1 and randomly alternating trial types (short: [2-inst, cnd1, gt:ra and ft:ra], cf. Section 2.1.1). For both monkeys the recording day of the selected dataset started late morning and lasted for nearly one hour and a half for monkey L, and one hour for monkey N. Table A.1 provides an overview of the recording day of the selected dataset from each monkey.

file content	**monkey L** filename	file size	**monkey N** filename	file size
cerebus configuration file	l101210-001.ccf	108.2 kB	i140703-001.ccf	187.1 kB
digital events, unsorted spikes	l101210-001.nev	287.7 MB	i140703-001.nev	168.3 MB
digital events, sorted spikes	l101210-001-02.nev	287.7 MB	i140703-001-03.nev	168.3 MB
signals of object sensors (N:+LFPdata)	l101210-001.ns2	8.5 MB	i140703-001.ns2	204.7 MB
raw neuronal signal	l101210-001.ns5	4.1 GB	i140703-001.ns6	5.8 GB
metadata	l101210-001.odml	2.9 MB	i140703-001.odml	2.5 MB
neuronal data annotated with metadata	l101210-001.mat		i140703-001.mat	

Table A.2 – Overview of datasets that will be published. Overview of content, name, and size for each file in the dataset of monkey L and N.

Each selected dataset was prepared for the publication in Scientific Data to contain the following files (see also Table A.2): The `ns5/ns6-file` contains the raw neural signals for monkey L and N, respectively. The `nev-file (original)` contains (i) the time stamps and spike waveforms of the

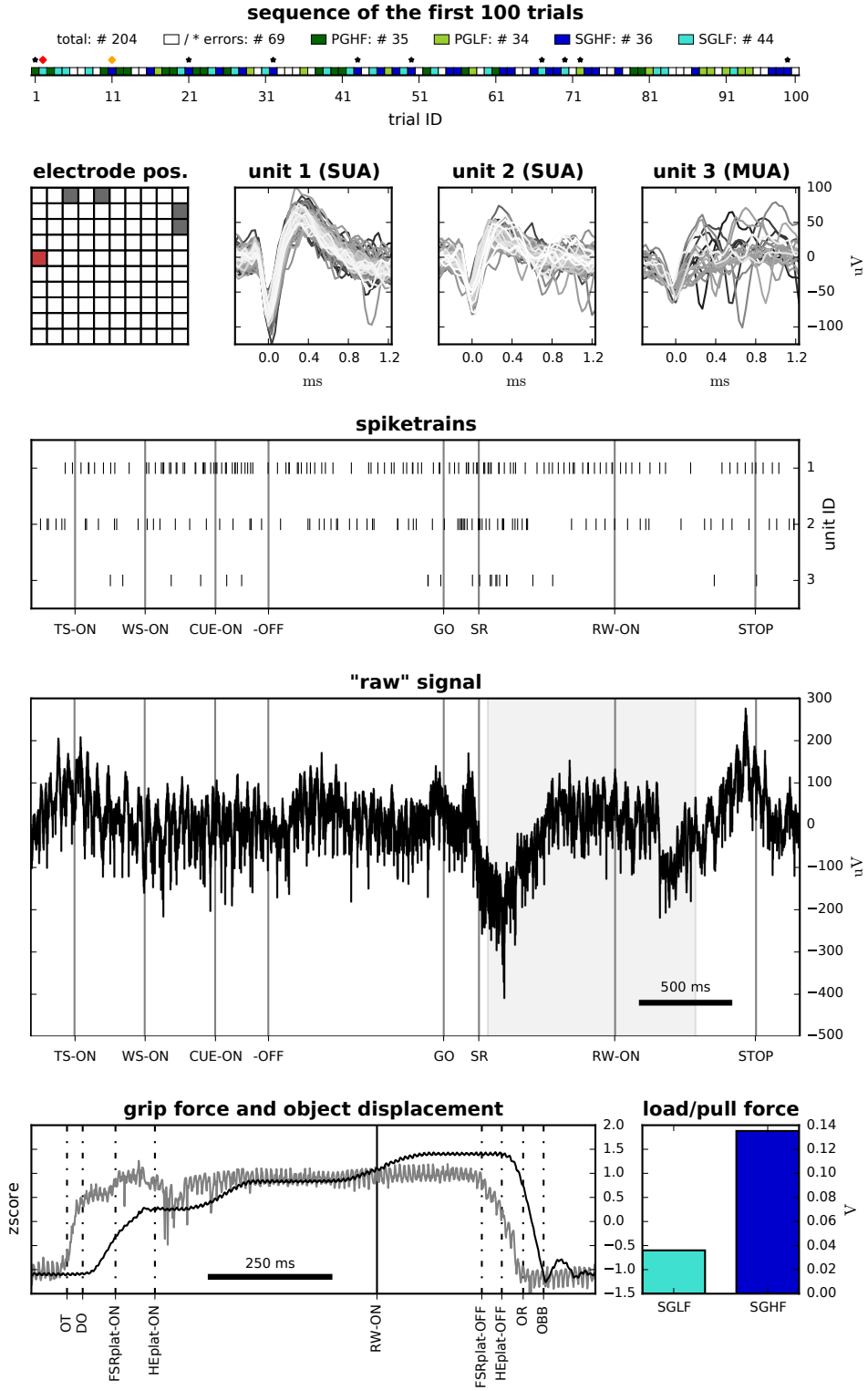

Figure A.1 – Overview of data types contained in l101210-001**.** The figure displays for monkey L the different data types contained in the selected dataset. Top panel: sequence of the first 100 trials (for trial types and errors see colour in legend) and the total number of trials (see # for correct, error, trial types in legend); red diamond marks selected trial for remaining data plots; orange diamond marks additionally selected trial to demonstrate load/pull force differences between the averaged load force signals in bottom right panel. Second row, left panel: position of selected electrode for the data plots. Second row, remaining panels: waveforms of three units from the selected electrode. Third row panel: spike trains of displayed units for the selected trial. Forth row panel: "raw" signal for the selected trial; grey shaded area marks time window of bottom left panel. Bottom left panel: grip force and object displacement signals for the selected trial. Bottom right panel: averaged load/pull force signals for the duration of the plateau of the grip force signal for the selected LF and HF trial. Important trial events are indicated as vertical lines in the corresponding data plots.

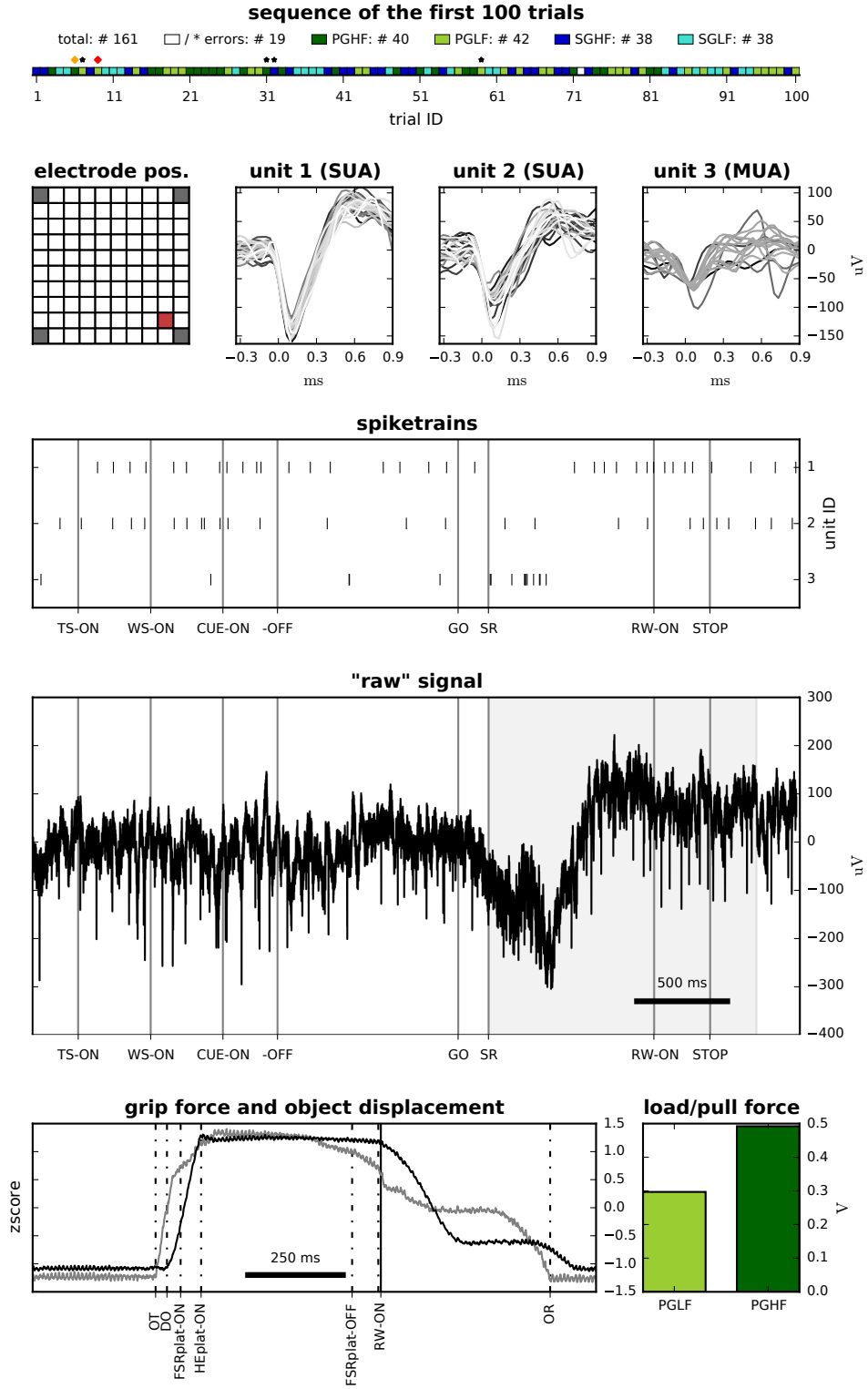

Figure A.2 – Overview of data types contained in i140703-001**.** The figure displays for monkey N(i) the different data types contained in the selected dataset. Top panel: sequence of the first 100 trials (for trial types and errors see colour in legend) and the total number of trials (see # for correct, error, trial types in legend); red diamond marks selected trial for remaining data plots; orange diamond marks additionally selected trial to demonstrate load/pull force differences between the averaged load force signals in bottom right panel. Second row, left panel: position of selected electrode for the data plots. Second row, remaining panels: waveforms of three units from the selected electrode. Third row panel: spike trains of displayed units for the selected trial. Forth row panel: "raw" signal for the selected trial; grey shaded area marks time window of bottom left panel. Bottom left panel: grip force and object displacement signals for the selected trial. Bottom right panel: averaged load/pull force signals for the duration of the plateau of the grip force signal for the selected LF and HF trial. Important trial events are indicated as vertical lines in the corresponding data plots.

online pre-sorted single and multi units, and (ii) the digital trial events reflecting the activation or deactivation of a physical device of the experimental apparatus. The `ccf-file` contains settings and configurations of Central Suite including the parameters used to extract the potential spike waveforms online. The `nev-file (spike-sorted)` was produced by the Plexon offline sorter. It equals the original nev-file, except for the unit IDs, which are replaced by the IDs of the offline sorted units. The `ns2-file` contains for both monkeys the analogue signals of the object sensors, and for monkey N, in addition, the LFP data (filtered and down-sampled version of the raw neural signals). The `odML-file` contains all metadata collected for the corresponding recording of the reach-to-grasp experiment. The `mat-file` contains the data of the the ns5/ns6-file, the spike-sorted nev-file and the ns2-file annotated with metadata from the odML-file. For examples of all data types see Figure A.1 for dataset from monkey L, and Figure A.2 for dataset from monkey N(i).

	# trials	# error trials		# correct trials				
	total	total	grip	total	SGLF	SGHF	PGLF	PGHF
monkey L	204	69	12	135	41	30	31	33
monkey N	160	19	16	141	35	35	35	36

Table A.3 – Overview of trials performed during the published datasets. Of the stated number of error trials, the monkey L and N used the wrong grip type in 12 and 16 trials, respectively. In the remaining error trials the monkeys initiated the movement too early. Trial types were altered randomly in the recordings which led to slightly different trial numbers for the different trial types.

The datasets and the corresponding code to access them (snapshot of the Python Neo package, snapshot of the Python odML package, and the custom-written `ReachGraspIO` extending the generic Neo `BlackrockIO` module) will be publicly available via the data portal of the German Neuroinformatics Node (G-Node) of the International Neuroinformatics Coordination Facility (INCF), called GNData[19].

			# SUA		# electrodes with	
	sorting ID	# MUA	total	SUA*	MUA/SUA	SUA
monkey L	*-02	49	93	89	86	65
monkey N	*-03	19	156	150	89	78

Table A.4 – Overview of offline sorted single and multi unit activity (SUA and MUA). For l101210-001 from monkey L it was possible to sort out 93 SUAs and 28 MUAs on 65 of 96 electrodes of the Utah-Array. On 21 additional electrodes with further MUAs were recorded. For i140703-001 from monkey N it was possible to sort out 156 SUAs and 8 MUAs on 78 of 96 electrodes of the Utah-Array. Here, 11 additional electrodes showed further MUAs. For monkey L, 89 of 93 identified SUAs had an SNR larger than 2.5 (SUA*). In monkey N, this was the case for 150 of 156 identified SUAs.

The recording from monkey N lasted with $16:43\,min$ several minutes longer than the recording from monkey L with only $11:49\,min$, monkey L ((cf. Table A.1). Nonetheless, monkey L executed 204 trials, while monkey N only performed 160 trials in total (cf. Table A.3), which may suggest that monkey N paused longer between each executed trial (longer inter-trial-intervals). However, monkey L performed only ~70% of all trials correctly, while monkey N successfully completed ~90% of all trials during the recording (cf. Table A.3). The high percentage of error trials in monkey L are mainly caused by an too early movement onsets reflecting the eagerness, but also the nervousness of the monkey L's character (cf. Section 2.2.1). In contrast to these error types, monkey L and monkey N performed a similar amount of incorrect grip types (12 and 16 times, respectively, cf. Table A.3). As

[19] http://g-node.github.io/g-node-portal/

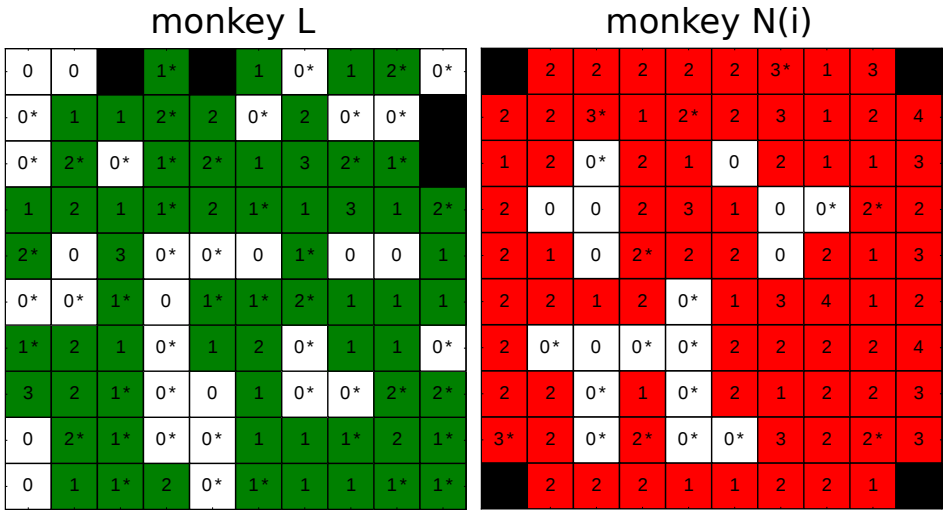

Figure A.3 – Overview of identified single and multi unit activities of l101210-001 **and** i140703-001**.** The figure displays the number of identified SUAs and MUAs across each array (numbers and * displayed for each electrode, respectively) for the two selected datasets for a data publication of monkey L and N(i).

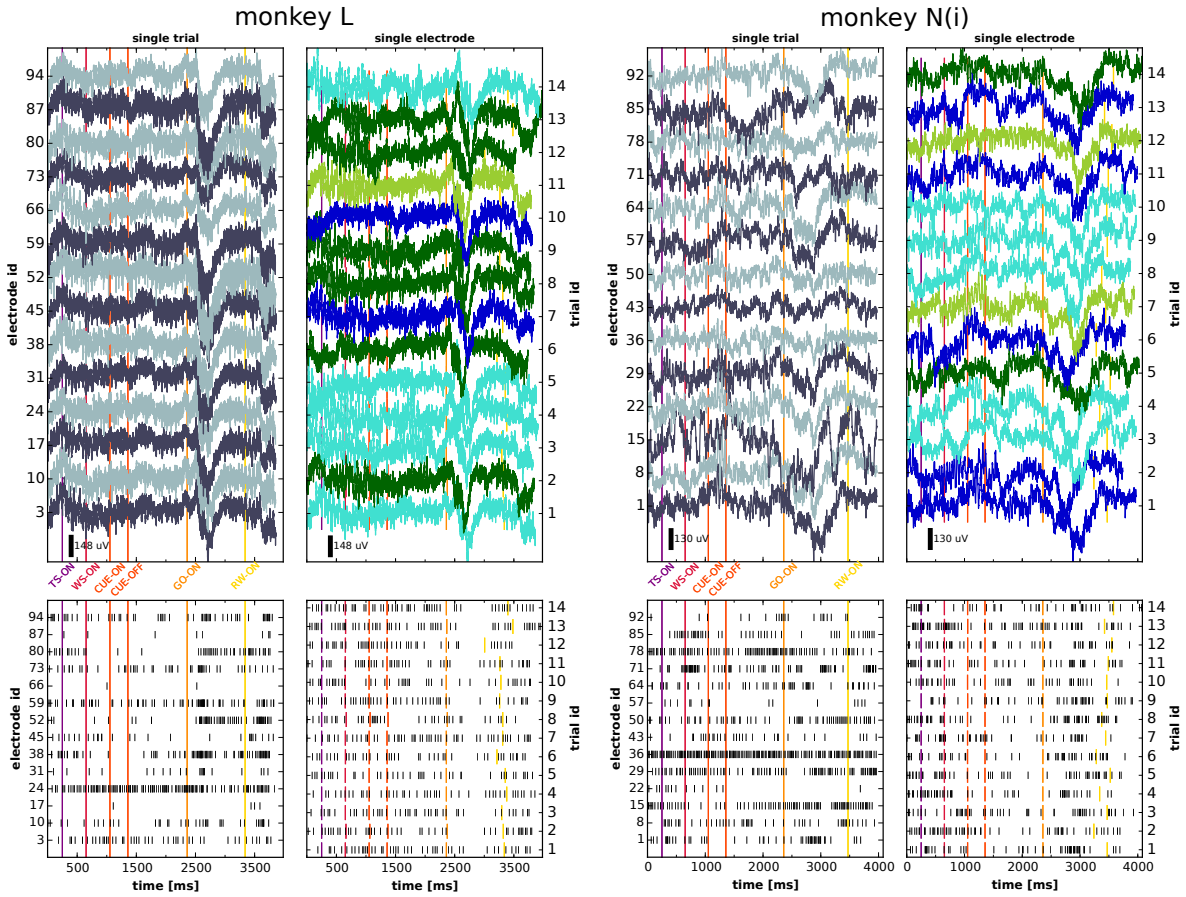

Figure A.4 – Overview of LFP and spike data of l101210-001 **and** i140703-001**.** The figure displays for monkey L (left columns) and monkey N(i) (right columns) for a single trial the LFP data, and the spike data from single units across a selection of electrodes (for each monkey: upper left, and bottom left panel, respectively), as well as the for a single electrode the LFP data and spike data from one single unit across a selection of correctly performed trials (for each monkey: upper right, and bottom right panel, respectively). Trial events (TS-ON, WS-ON, CUE-ON, CUE-OFF, GO-ON, and RW-ON) are indicated as coloured vertical lines in each plot. For each monkey: trial types of selected trials in upper right panels are indicated as colour (SGHF: dark blue; SGLF: cyan; PGHF: dark green; PGLF: light green).

previously stated, for both monkeys the trial types alternated randomly in the trial sequence, which led to slightly different numbers of trials with the same trial type in the each dataset (cf. Table A.3).

The quality of the spiking activity in the selected datasets was high, which resulted in a relatively robust offline spike sorting with high numbers of SUA and MUA distributed over all electrodes of the array (cf. Table A.4 and Figure A.3). The quality assessment analysis of the identified SUAs confirmed the high quality of recorded spike data, revealing a high numbers of SUA with a SNR larger than 2.5 for both datasets (cf. #SUA* in Table A.4). For details on how the offline spike sorting and corresponding quality assessment was performed see Section 2.4.3 and Section 2.4.4.4, respectively.

Preliminary study on beta profiles

For a poster presentation at the Society for Neuroscience conference 2014, I performed a preliminary study, in which I tried to relate the monkey's task engagement to the corresponding amplitude modulation differences in the beta-oscillations (*"Latency of LFP beta power peak during movement preparation correlates with reaction time"*, Zehl et al., 2014a). Note that at the time this preliminary study took place (November 2014) neither the implementation of the odML metadata framework nor the implementation of the Neo data framework was completed yet. The analysis was performed using preliminary implementations of the frameworks, which comprised less functionality. For example, at that time, it was not possible to extract error trials from the data.

Background: Beta oscillations (15–30 Hz) are a prominent feature of neuronal population signals recorded in the sensorimotor cortex during motor behaviour (Kilavik et al., 2013). They have been extensively studied in instructed delay tasks in which a cue provides prior information about the movement to be performed after a GO signal presented at the end of a delay period. In such tasks, beta power often exhibits an initial peak of high power shortly after the cue presentation, then decreases during the delay and is lowest during movement execution. It was proposed that the pre-movement decrease in beta power is related to an increased activity in motor cortex related to the upcoming movement execution (e.g., Kilner et al., 2005). Thus, I aimed to test the hypothesis whether the temporal profile of beta power modulations during movement preparation correlates with behavioural measures of the movement, such as the reaction time (RT).

Data selection: As described in Section 2.2.1, each monkey performed differently in the R2G experiment. Nonetheless, for the analysis of the preliminary study I wanted to compare datasets, in which the monkeys were similarly engaged in the task. For this reason, I followed four steps to select datasets and one step to select trials for the beta profile analysis procedure (cf. below, and corresponding methods in Section 4.2).

In the first step, I identified all datasets for monkey T(t), monkey L, and monkey N(i) available at that time (November 2014), in which the monkeys had to perform the standard reach-to-grasp task with either condition setting 1 or condition setting 2 with a random trial type sequence (2-inst, cnd1 or cnd2, gt:ra and ft:ra, cf. Section 2.1.1). Note that for monkey T(t) and monkey L only datasets with LFP data stored in the ns2-file were used to avoid additional implementation of code to extract LFP data from ns5-files (cf. Section 2.3.3).

In the second step, I excluded datasets with less than 80 correctly performed trials for the grip-first condition and with less than 40 correctly trials for the force-first condition (cf. cnd1 and cnd2, respectively in Figure A.5), because a too low number of correct trials indicates a low motivation of the monkey to perform the requested task. The decision to use different minimum trial numbers for

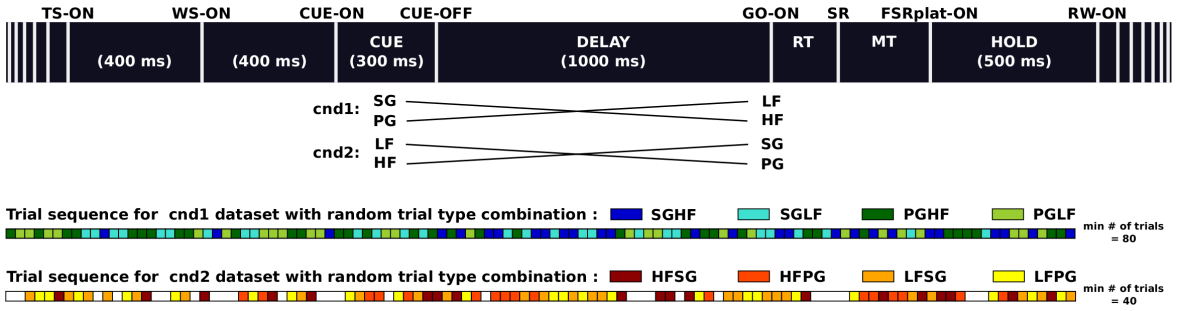

Figure A.5 – Data selection for preliminary study based on task and trial numbers. The figure displays the trial scheme of the 2-inst task paradigm (top) and an example of the trial type sequence of the selected condition (cnd) settings (bottom). Only cnd1 datasets with at least 80 trials and cnd2 datasets with at least 40 trials were considered. White spots in example trial sequence in cnd2 indicate error trials. TS-ON: trial start, WS-ON: central LED turns on, CUE-ON: cue turns on, CUE-OFF: cue turns off, GO-ON: GO signal turns on, RT: reaction time, SR: switch release (movement start), MT: movement time, FSRplat-ON: rough start point of a constant object hold by the monkey, RW-ON: reward pump turns on, SG: side grip, PG: precision grip, LF: low force, HF: high force.

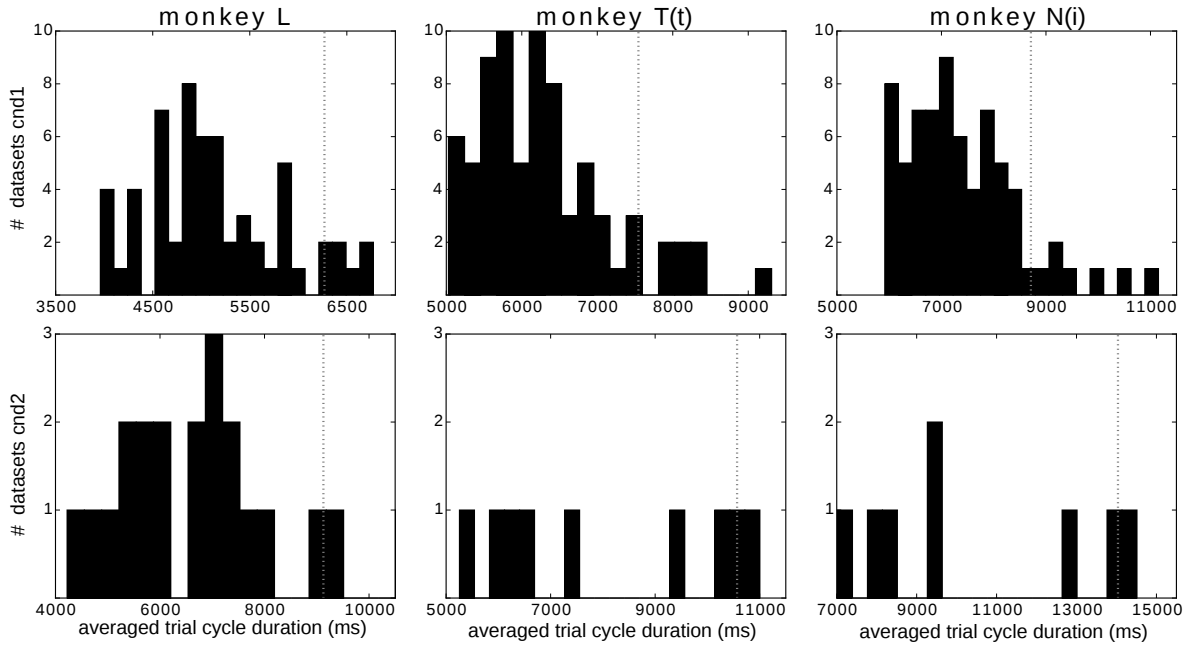

Figure A.6 – Data selection for preliminary study based on averaged trial cycle. The figure displays the distribution of averaged trial cycle duration of all cnd1 datasets with more than 80 trials (first row) and cnd2 datasets with more than 40 trials (second row) for monkey L, T(t), and N(i) (first, second, and this column, respectively). The trial cycle is is defined by the averaged duration from one trial start (TS-ON_i) to the next (TS-ON_{i+1}) of correctly performed trials. The grey dotted vertical lines indicate in each distribution plot the trial cycle limit for further selecting datasets for the analysis of the preliminary study.

cnd1 and cnd2 datasets is explained by the fact that, on average, the monkeys conducted less total trial numbers and committed more error trials in cnd2 than in cnd1 tasks.

In the third step, I further excluded datasets with extraordinary long averaged trial cycle, which was defined by the averaged duration from one trial start (TS-ON$_i$) to the next (TS-ON$_{i+1}$) of correctly performed trials (cf. Figure A.6). In datasets with extraordinary long averaged trial cycle the monkey paused for a long time between trials (long inter-trial-intervals) and/or the monkey conducted many error trials, which I considered both as indications for a poor engagement of the monkey in the requested task of the corresponding recording. The distribution of averaged trial cycle durations for the datasets in Figure A.6 not only displays the cut-off duration of extraordinary long averaged trial cycle, but also illustrates the general performance differences between the task conditions (cnd1 is preferred over cnd2 in all monkeys) and between the monkeys (monkey L performed better than monkey T(t) which in turn performed better than monkey N(i)).

In the fourth step, I generated a power spectrum of the complete recording averaged across all electrodes to further exclude datasets with strong heart beat oscillation in their power spectrum. After this step, 53, 65, and 63 datasets with cnd1, and 19, 16, and 7 datasets with cnd2 remained for monkey L, T(t), and N(i), respectively.

Subsequently to this step, I identified the dominant beta frequency band for each selection in the corresponding dataset-average of the power spectra (beta peak frequency $\pm 2.5\,Hz$). The dataset-averaged of the power spectra revealed a peak beta frequency of $21\,Hz$, $17.5\,Hz$, and $16.5\,Hz$ for the dataset selection of cnd1, and $20.5\,Hz$, $17\,Hz$, and $16\,Hz$ for the dataset selection of cnd2 for monkey L, T(t), and N(i), respectively. Further investigations, which I conducted after this preliminary study, showed that the difference in peak frequencies is actually not monkey specific, but mainly depending on the relation of time in the recording in which a monkey is performing the task (trials) or not (inter-trial-intervals). During a trial all monkeys usually show a peak beta frequency around $21\,Hz$, while outside of a trial the peak beta frequency is considerably lower (around $16\,Hz$). Following this argumentation, the lower beta peak frequencies of monkey T(t) and N(i) actually revealed that these two monkeys spent on average more time outside of a trial while conducting a recording than monkey L. Nonetheless, for the preliminary study, the monkey-specific beta frequency bands were chosen for calculating the corresponding beta profiles (cf. below, and corresponding methods in Section 4.2).

In the last step of the data selection, I created RT-specific trial selections for SG, and PG, as well as, LF, and HF trials from the chosen datasets of cnd1 and cnd2 (cf. Figure A.8). Based on the signal variance, signals of some electrodes and some trials were additionally excluded from the analysis during this selection step (cf. LFP quality assessment in Section 2.4.4.3). The four resulting trial selections were then used to finally calculate the trial averaged beta profiles for trials with slow, median, or fast RT (cf. Figure A.9 and Figure A.10). The number of trials selected for each dataset selection are listed in Table A.5.

	# cnd1 trials with ...						**# cnd2 trials with ...**					
	fast RTs		median RTs		slow RTs		fast RTs		median RTs		slow RTs	
	SG	PG	SG	PG	SG	PG	LF	HF	LF	HF	LF	HF
monkey L	702	805	695	842	935	545	318	153	237	284	207	314
monkey T(t)	1187	329	697	813	508	937	67	62	59	71	48	70
monkey N(i)	811	496	656	615	394	820	53	49	57	46	44	54

Table A.5 – Overview of selected trials for the trial-averaged beta profile analysis. The table summarizes the numbers of selected trials with fast, median, and slow RTs for each dataset selection of monkey L, T(t), and N (i), split into grip types (SG and PG) for cnd1 recordings and force types (LF and HF) for cnd2 recordings.

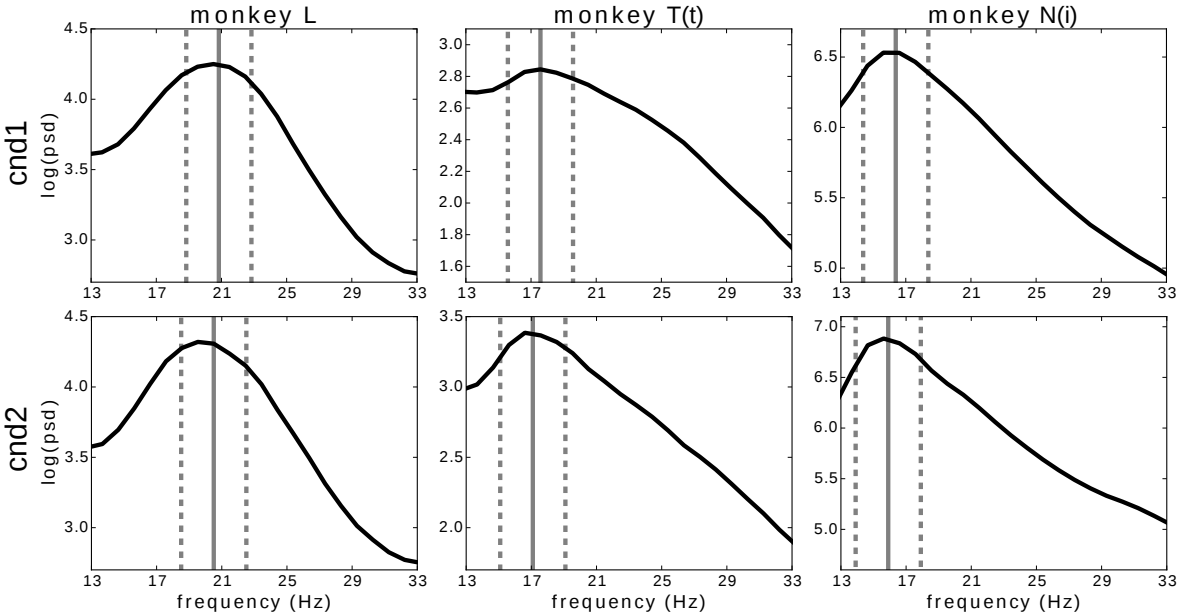

Figure A.7 – Data selection for preliminary study based on signal quality. For the beta profile analysis the analogue signals needed to be filtered to a beta frequency range. The figure shows the power spectrum of the complete recording time averaged across electrodes and all selected datasets for each monkey. Datasets with strong heart beat oscillation in their power spectrum were detected prior to the averages and excluded from the subsequent profile analysis. The beta peak frequencies (grey solid vertical lines) were identified in each plot and the corresponding dominant beta frequency bands (beta peak frequency $\pm 2.5\,Hz$, grey dotted vertical lines) were defined for calculating the beta profiles for each data selection.

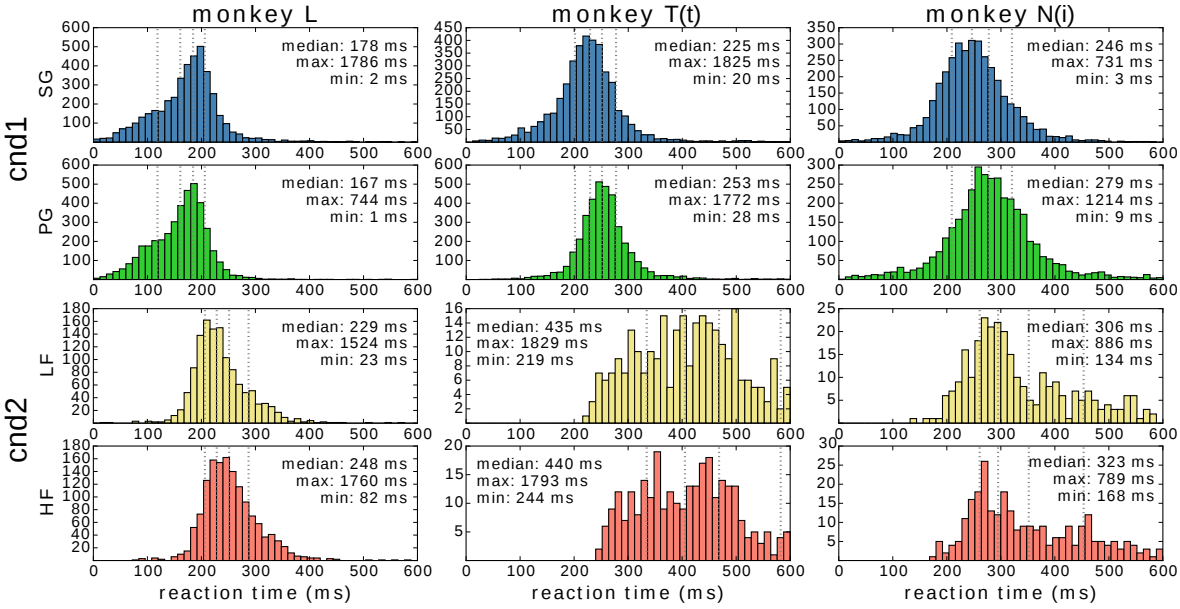

Figure A.8 – Data selection for preliminary study based on reaction times. The figure displays the distribution of reaction times (RT) for all trials split into SG (first row, blue) and PG (second row, green) for cnd1 and LF (third row, yellow) and HF (forth row, red) for cnd2 of the selected datasets (cf. Figure A.5 and Figure A.6) for monkey L, T(t), and N(i) (first, second, and this column, respectively). For averages in the beta profile analysis, the first fifth of the RT sorted trials with lowest RT were selected as ***fast***, the third fifth as ***median***, and the last fifth with longest RTs as ***slow*** trials. RT class limits are indicated by grey dotted vertical lines. In addition, the median, maximum, and minimum RT of each distribution is stated in each plot.

Independent from the trial selection, one can also conclude from the RT distributions in Figure A.8 that monkey L was generally able to perform the trials faster than monkey T(t) and N(i), reflecting her overall eagerness to work. One can further see that, on average, the RTs are clearly faster in trials from cnd1 datasets compared to trials from cnd2 datasets in monkey L and T(t), but only slightly faster in monkey N(i). This fact, in combination with the report of the experimenter that for monkey N(i) the number of grip errors conducted in cnd2 recordings were extraordinarily high, suggests that, in contrast to monkey N(i), monkey L and T(t) knew that they had to wait for the GO-ON signal to learn about the requested grip type in cnd2 recordings. For this reason, monkey L and T(t) had longer RTs in cnd2 as compared to cnd1 in which the grip was already revealed in the CUE period allowing the monkeys to anticipate the GO-ON signal to initiate the movement as early as possible. Instead, monkey N(i) seemed to not have fully learned that the GO-ON signal provides further information for the requested behaviour.

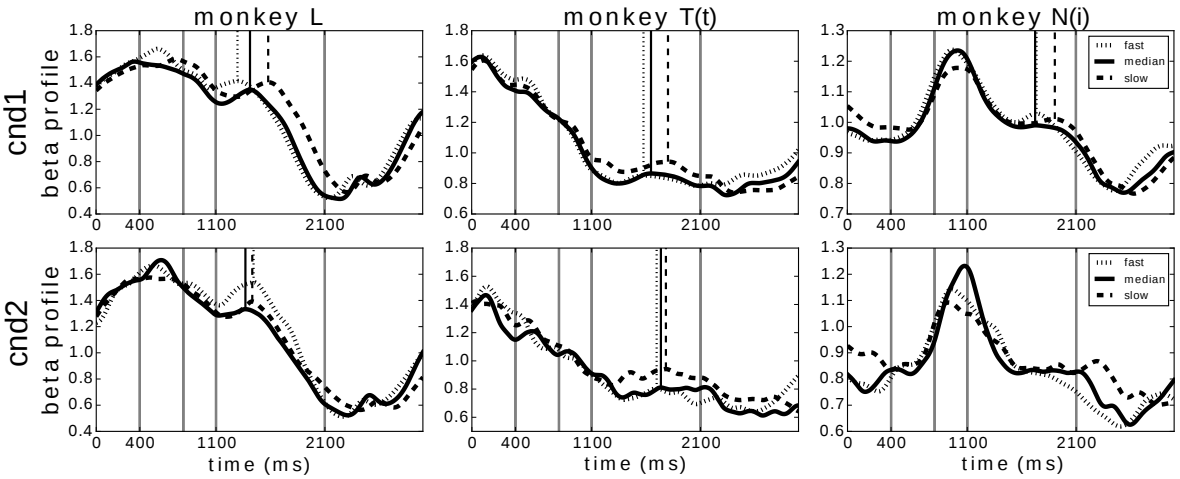

Figure A.9 – Beta profile peak in preparatory delay period is shifted with increasing RT. The figure shows the trial-averaged profile for each RT class (fast, median, slow, see legend in right panels) in each monkey L, T(t), and N(i) for trials from cnd1 datasets (first row) and cnd2 datasets (second row). The power peaks of the profiles in the delay period are indicated with vertical lines, emphasizing that the time point of the peak is delayed with increasing RTs in all three monkeys. Gray solid lines: trial events (WS-ON, CUE-ON, CUE-OFF, and GO-ON)

Beta profiles of selected data: For the beta profile analysis in each monkey, each electrode signal was filtered according to the beta frequency bands determined in Figure A.7 and normalized (z-scored). In a second step, the amplitude profile of each signal was calculated using a Hilbert transformation, and averaged across all electrodes. Subsequently, the signals of the selected trials were extracted and averaged according to their RT class (fast, median, or slow) and/or grip or force type (SG and PG for cnd1 datasets; LF and HF for cnd2 datasets).

Figure A.9 shows the trial-averaged beta profile for each RT class (fast, median, slow) in each monkey for trials from cnd1 datasets (first row) and cnd2 datasets (second row), independent of grip and force type. Focussing on the power peaks of the beta profiles in the preparatory delay period, the time point of the peak in the trial-averages from cnd1 datasets is detected earliest for fast RTs, then for median RTs, and latest for slow RTs in all three monkeys. In contrast, the time point of the peak in the trial-averages from cnd2 datasets remains similar between the RT classes.

Compared to Figure A.9, Figure A.10 displays the trial-averaged beta profile for each RT class (fast, median, slow) in each monkey split into SG and PG trials from cnd1 datasets, and LF and HF trials form cnd2 datasets. Focussing on the power differences of the beta profiles in the preparatory

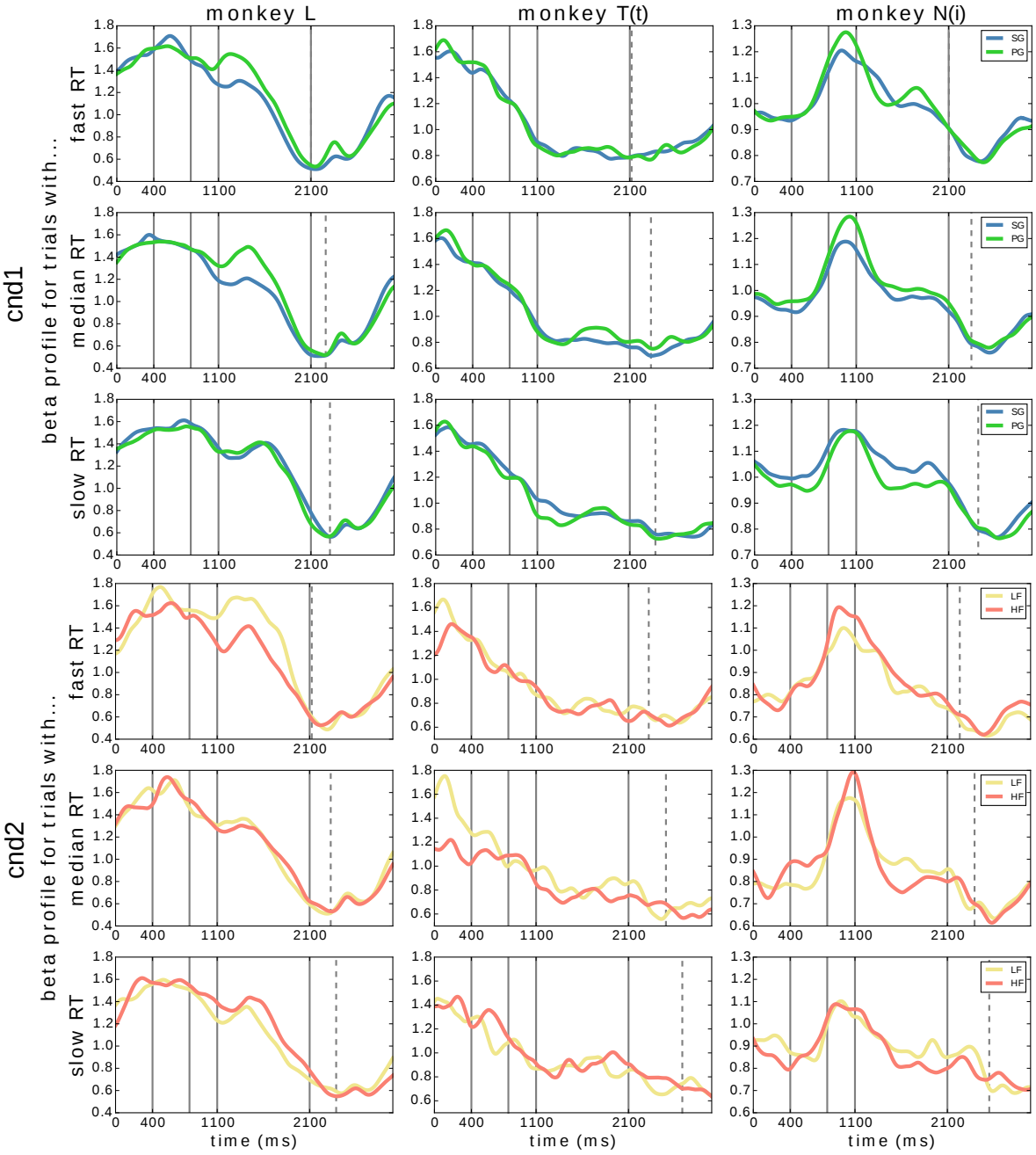

Figure A.10 – Beta profile differences in preparatory delay period for grip and force types. The figure displays the trial-averaged profiles split into grip and force types for cnd1 and cnd2 datasets (first - third and fourth to sixth row) for fast, median, and slow RT class trials for monkey L, T(t), and N(i), respectively. SG: blue profiles; PG: green profiles; LF: yellow profiles; HF: red profiles. Gray solid lines: trial events (WS-ON, CUE-ON, CUE-OFF, and GO-ON); Gray dashed line: minimum RT for the corresponding RT class.

delay period, the results are inconclusive. Nonetheless, for the cnd1 datasets, the power of the beta profiles tend to be larger for PG than for SG in trials with fast or median RTs during the early delay period for monkey L, the late delay period for monkey T(t), and the CUE or late delay period for monkey N(i). Moreover, in trials with slow RTs there is no observable difference in monkey L, and even a reversed difference in the early delay in monkey T(t), and the delay in monkey N(i). In contrast, for the cnd2 datasets, the power of the beta profiles tend to be similar between LF and HF trials independent of the RT class, except for the fast RT class in monkey L, where the power of the beta profiles tend to be larger in LF than in HF trials. Nevertheless, the results of the power differences in the trial-averaged profiles of grip and force types need to be treated carefully, not only because of the differences in trial numbers from which the averages were calculated (cf. Table A.5), but also because the single trial profiles are highly variable (cf. Figure A.11) indicating that the amplitude modulation of beta oscillations is affected by more than just the reaction time (RT).

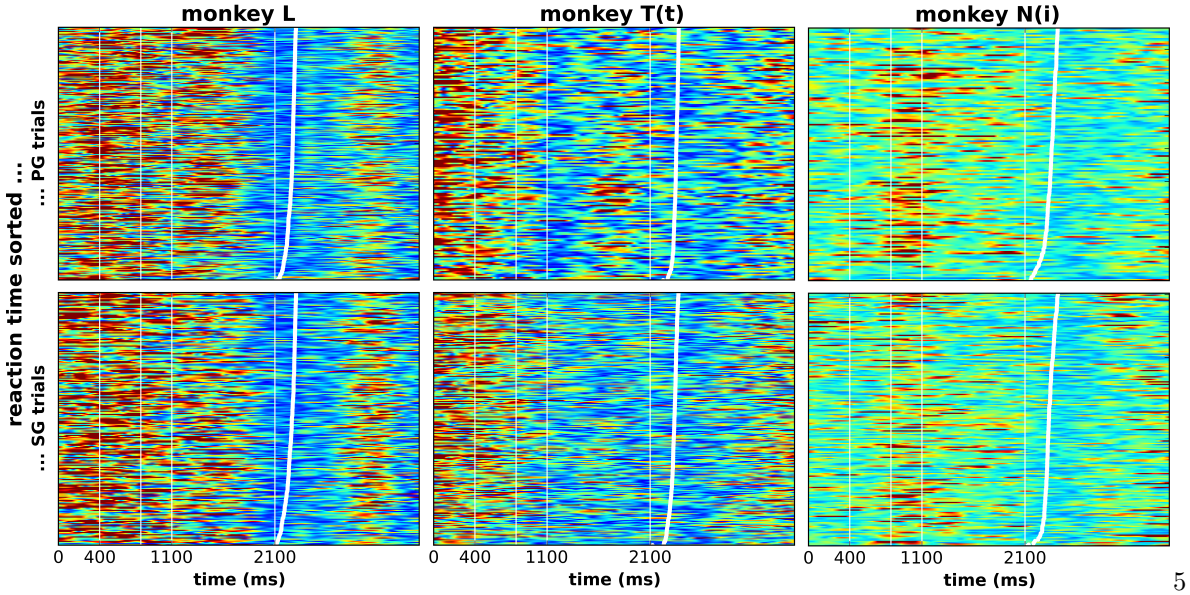

Figure A.11 – **Example of trial-by-trial variability of beta profile analysis.** The figures displays the best trials (RT $<$ 200 ms) sorted by RT (bottom–top: fast–slow) extracted from the PG and SG trials (top and bottom panels, respectively) of five datasets from monkey L, T(t), and N(i) (left, middle and right column, respectively). White thin vertical lines: trial events (WS-ON, CUE-ON, CUE-OFF, and GO-ON). White fat curved line: RT of each trial. Colour code: red (high power), blue (low power).

Summary, conclusions and outlook: The timing of the beta power peak and strong subsequent decrease during the delay period changes in relation with mean RT if the grip type is known in advance (cnd1), and is not affected by RT if the force type is known in advance (cnd2). Moreover, the amplitude of the beta peak power during the delay period may depend on the grip type if the grip type is known in advance (cnd1), but seems not to depend on knowing the force type in advance (cnd2). From these results I speculated the following: if the grip type is known in advance, the corresponding movement is prepared in advance, and this is reflected by an earlier beta modulation in the delay period. In addition, if RT is taken as a measure of how well a monkey was prepared for the upcoming movement, the difference in amplitude during the delay period reflects the difference in preparing for the different grip types. The difficulty of interpreting the results of the beta profile analysis is based the fact that they emerge from power modulations on a trial-by-trial basis. For this reason, if the trials are wisely chosen for the corresponding averages, a dependency of the beta modulation is detectable,

although there is a high trial-by-trial variability (cf. Figure A.11). Nevertheless, if I assume that the beta power modulation reflects how well a monkey is prepared for the upcoming movement, a high trial-by-trial variability is to be expected depending on the current readiness for work of the monkey.

For this reason, one should focus on a trial-by-trial analysis for future studies to relate the monkeys behaviour in time directly with the corresponding beta profile modulation and make use of more metadata criteria to better define the behavioural conditions the monkey is working with during a recording (e.g., adaptation to trial type repetitions or irritation through errors).

Overview of offline spike sorted datasets

The following tables summarize the results of the offline spike sorted datasets that were performed until Wednesday 30th November, 2016. The datasets are listed by the common filename. The file appendix (appx) used for the nev-file containing the offline spike sorting results are listed separately. For details on the offline spike sorting procedure see Section 2.4.3. In addition, the tables state the task paradigms conducted in each of the datasets (for details see Section 2.1) and the used condition (see Table 2.2 for the cnd codes). Besides the number of single and multi units (#SUA and #MUA), the tables also indicate the results of the quality assessment of the spike data, by stating the number of SUAs with a signal to noise ratio larger than 2.5 (#SUA*). For details on the procedure of the quality assessment of the spike data see Section 2.4.4.4. Furthermore, the tables state for each dataset if the raw data (ns5 or ns6) are available. This is an interesting fact, because with available raw data it is not only possible to resort the spiking data offline, but also to redo the extraction of potential spike waveforms (cf. Section 2.3.3). Further abbreviations used in the tables: (gt:blX: grip types alternate in blocks of X trials; ft:blX: force types alternate in blocks of X trials; n.y.i.: not yet identified).

filename	appx	task	cnd	raw data	# SUA	# SUA*	# MUA
t190310-002	-02	2-inst	11, gt:bl3	no	16	n.y.i.	26
t020410-001	-02	1-inst	11	no	21	n.y.i.	18
t140410-001	-02	2-inst	1	no	10	n.y.i.	21
t150410-001	-02	2-inst	1	no	22	8	21
t150410-002	-03	2-inst	1	no	20	n.y.i.	21
t230410-001	-02	2-inst	1	no	25	11	24
t290410-001	-02	2-inst	1	no	23	6	21
t180610-001	-02	2-inst	1	no	17	3	25
t060710-003	-02	2-inst	1	no	12	2	16
t070710-002	-02	2-inst	1	no	11	2	21
t100910-002	-02	2-inst	2, gt:bl4	no	14	n.y.i.	18
t130910-001	-02	2-inst	2, gt:bl5	no	19	n.y.i.	23
t140910-001	-02	2-inst	2	no	18	n.y.i.	21
t150910-001	-02	2-inst	2	no	19	n.y.i.	19
t160910-001	-02	2-inst	2	no	19	n.y.i.	19
t170910-002	-02	1-inst	2	no	18	n.y.i.	18
t280910-001	-02	1-inst	1	no	9	1	22
t300910-001	-02	1-inst	1	no	6	n.y.i.	19
t300910-002	-02	1-inst	1	no	10	n.y.i.	20
t300910-003	-02	1-inst	1	no	12	2	20
t011010-001	-02	1-inst	1	no	18	2	21
t011010-002	-02	1-inst	1	no	18	n.y.i.	21
t041010-001	-02	1-inst	1	no	18	n.y.i.	21
t041010-003	-02	1-inst	1	no	19	n.y.i.	21
t151010-001	-02	1-inst	1	no	30	9	28
t151010-002	-02	1-inst	14	no	30	n.y.i.	28
t140211-001	-02	map	-	no	10	n.y.i.	15

Table A.6 – Overview of offline sorted datasets of the first recording period of monkey T.

filename	appx	task	cnd	raw data	# SUA	# SUA*	# MUA
a110628-002	-01	map	-	no	35	n.y.i.	15
a110629-001	-02	2-inst	141	no	30	n.y.i.	17
a110705-001	-02	2-inst	131	no	34	n.y.i.	26
a110705-002	-02	2-inst	131	no	33	n.y.i.	25
a110706-001	-01	2-inst	131	no	33	n.y.i.	24
a110706-002	-01	2-inst	141	no	33	n.y.i.	25
a110706-003	-01	2-inst	11	no	33	n.y.i.	25
a110706-004	-02	map	-	no	34	n.y.i.	23
a110707-001	-01	2-inst	11	no	33	n.y.i.	26
a110711-001	-01	2-inst	11, gt:bl3	no	27	n.y.i.	21
a110712-001	-02	2-inst	11, gt:bl3	no	31	n.y.i.	23
a110718-001	-02	2-inst	11, gt:bl3	no	34	n.y.i.	27
a110719-001	-02	2-inst	11	no	40	n.y.i.	19
a110720-001	-02	2-inst	11	no	39	n.y.i.	15
a110721-001	-02	2-inst	11	no	41	n.y.i.	15
a110816-001	-01	map	11	no	60	n.y.i.	15
a110818-001	-02	2-inst	11	no	50	n.y.i.	6
a110824-001	-02	2-inst	11	no	50	n.y.i.	8
a110825-001	-02	2-inst	1	no	53	n.y.i.	8
a110826-001	-02	2-inst	1	no	52	n.y.i.	12
a110901-002	-02	2-inst	1	no	56	n.y.i.	12
a110902-001	-02	2-inst	1	no	59	n.y.i.	21
a110905-001	-02	2-inst	1	no	50	n.y.i.	12

Table A.7 – Overview of offline sorted datasets of the second recording period of monkey T.

filename	appx	task	cnd	raw data	# SUA	# SUA*	# MUA
l110203-005	-03	map	-	no	85	n.y.i.	96
l110204-002	-04	map	-	no	90	n.y.i.	78
l110207-001	-04	map	-	no	91	n.y.i.	79
l110208-001	-04	2-inst	1	no	95	75	81
l110208-003	-04	2-inst	2	no	96	75	82
l110208-006	-04	map	-	no	103	n.y.i.	68
l110209-001	-06	2-inst	1	no	80	69	82
l110209-002	-06	2-inst	2	no	79	66	74
l110404-001	-05	2-inst	1	no	72	61	50
l110404-002	-06	2-inst	14	no	74	n.y.i.	49
l110404-003	-03	2-inst	14	no	67	n.y.i.	44
l110405-001	-06	2-inst	14	no	67	n.y.i.	40
l110405-002	-05	2-inst	14	no	69	n.y.i.	39
l110407-001	-06	2-inst	14	no	73	n.y.i.	47
l110407-002	-06	2-inst	14	no	73	n.y.i.	47
l110408-002	-05	2-inst	1	no	69	56	53
l110411-001	-04	2-inst	1	no	84	74	43
l110411-002	-03	2-inst	13	no	84	n.y.i.	43
l110412-001	-05	2-inst	13	no	87	n.y.i.	32
l110412-002	-03	2-inst	13	no	85	n.y.i.	34
l110412-004	-03	2-inst	23	no	87	n.y.i.	31
l110413-001	-03	2-inst	13	no	82	n.y.i.	32
l110413-002	-03	2-inst	13	no	82	n.y.i.	33
l110413-004	-04	2-inst	23	no	80	n.y.i.	34
l110414-002	-04	2-inst	13	no	83	n.y.i.	29
l110414-003	-04	2-inst	13	no	83	n.y.i.	28
l110414-004	-04	2-inst	23	no	83	n.y.i.	28
l110415-002	-03	2-inst	1	no	86	79	27
l110415-004	-04	2-inst	2	no	87	80	28

Table A.8 – Overview of offline sorted datasets of monkey L (year 2011).

filename	appx	task	cnd	raw data	# SUA	# SUA*	# MUA
l101005-002	-10	2-inst	1	no	102	n.y.i.	88
l101005-004	-02	2-inst	1	no	68	n.y.i.	75
l101006-001	-03	2-inst	1	no	63	53	82
l101006-002	-03	2-inst	1	no	64	n.y.i.	68
l101006-003	-03	2-inst	1	no	67	72	76
l101007-001	-02	2-inst	1	no	78	n.y.i.	78
l101007-002	-05	2-inst	1	no	77	n.y.i.	72
l101007-003	-03	2-inst	1	no	66	84	79
l101008-003	-01	2-inst	1	no	89	n.y.i.	78
l101008-004	-04	2-inst	1	no	68	n.y.i.	79
l101008-005	-01	2-inst	1, ft:bl20	no	81	n.y.i.	79
l101013-002	-02	2-inst	1	no	103	85	60
l101013-003	-02	2-inst	1	no	77	n.y.i.	69
l101013-004	-03	2-inst	2	no	95	81	65
l101013-005	-02	2-inst	2	no	88	n.y.i.	71
l101013-006	-02	2-inst	1	no	84	n.y.i.	74
l101014-002	-04	2-inst	1, ft:bl10	no	101	81	75
l101014-003	-03	2-inst	11, gt:bl10	no	85	n.y.i.	60
l101014-004	-04	2-inst	11, gt:bl10	no	98	n.y.i.	63
l101014-005	-04	2-inst	13, ft:bl10	no	120	n.y.i.	68
l101014-006	-04	2-inst	1	no	77	n.y.i.	86
l101014-007	-03	2-inst	13, ft:bl1	no	93	n.y.i.	65
l101015-001	-04	2-inst	1	no	110	75	82
l101015-002	-05	2-inst	2	no	111	73	85
l101015-003	-03	2-inst	2	no	79	n.y.i.	74
l101015-004	-02	2-inst	14, ft:bl10	no	89	n.y.i.	61
l101015-005	-03	2-inst	13	no	66	n.y.i.	55
l101015-006	-03	2-inst	13	no	44	n.y.i.	40
l101015-007	-05	2-inst	14	no	87	n.y.i.	50
l101015-008	-02	2-inst	14	no	76	n.y.i.	64
l101105-002	-05	2-inst	1	no	56	n.y.i.	60
l101106-001	-05	2-inst	1	no	65	n.y.i.	58
l101106-003	-02	2-inst	2	no	64	n.y.i.	58
l101108-001	-03	2-inst	1	no	83	49	90
l101108-002	-04	2-inst	2	no	83	48	91
l101109-001	-06	2-inst	1	no	84	n.y.i.	80
l101110-003	-04	2-inst	1	no	86	61	73
l101110-005	-08	2-inst	2	no	84	54	75
l101111-002	-04	2-inst	1	no	86	54	84
l101111-003	-04	2-inst	2	no	86	54	83
l101111-004	-04	2-inst	24	no	86	n.y.i.	85
l101111-005	-05	2-inst	23	no	86	n.y.i.	85
l101126-002	-02	2-inst	1	no	106	86	65
l101126-003	-04	2-inst	2	no	105	77	72
l101202-001	-06	2-inst	1	no	102	92	57
l101202-002	-05	2-inst	2	no	103	94	60
l101202-004	-04	1-inst	13, ft:bl10	no	105	n.y.i.	60
l101202-005	-04	1-inst	14, ft:bl10	no	103	n.y.i.	59
l101206-005	-02	sleep	-	no	136	n.y.i.	35
l101208-002	-04	2-inst	1	yes	100	95	53
l101209-001	-05	2-inst	1	no	107	98	71
l101209-002	-04	2-inst	13	no	109	n.y.i.	71
l101209-004	-04	2-inst	14	no	107	n.y.i.	72
l101210-001	-02	2-inst	1	yes	93	89	50
l101213-002	-03	2-inst	1	yes	105	102	58
l101213-005	-02	sleep	-	yes	135	n.y.i.	58
l101216-002	-05	2-inst	1	no	119	94	78
l101216-004	-04	2-inst	2	no	119	90	77
l101220-002	-04	2-inst	1	no	107	89	84
l101220-003	-03	2-inst	2	no	107	90	84

Table A.9 – Overview of offline sorted datasets of monkey L (year 2010).

filename	appx	task	cnd	raw data	# SUA	# SUA*	# MUA
n130509-001	-02	2-inst	1	no	38	n.y.i.	42
n130510-001	-03	2-inst	1	no	44	n.y.i.	39
n130516-001	-03	2-inst	1	no	55	n.y.i.	29
n130516-003	-04	2-inst	1	no	55	n.y.i.	28
n130517-003	-03	2-inst	1	no	53	n.y.i.	40
n130523-001	-04	2-inst	1	no	44	n.y.i.	54
n130524-001	-03	2-inst	1	no	52	n.y.i.	50
n130530-001	-03	2-inst	1	no	57	n.y.i.	43
n130613-001	-03	2-inst	1	no	66	n.y.i.	42
n130614-003	-07	2-inst	1	no	79	n.y.i.	39
n130619-002	-04	2-inst	1	no	29	n.y.i.	77
n130620-002	-03	2-inst	1	no	27	n.y.i.	76

Table A.10 – Overview of offline sorted datasets of first recording period of monkey N.

filename	appx	task	cnd	raw data	# SUA	# SUA*	# MUA
i140613-001	-04	2-inst	-	no	154	148	12
i140614-004	-07	map	-	no	150	147	11
i140615-002	-05	rest	1	no	135	134	12
i140616-001	-04	2-inst	1	no	161	157	19
i140617-001	-05	2-inst	1	no	167	159	15
i140617-002	-03	iso	1	no	163	160	14
i140627-001	-05	2-inst	1	yes	160	155	16
i140627-002	-04	iso	1	yes	160	157	13
i140701-001	-05	2-inst	1	yes	150	148	9
i140701-002	-05	iso	1	yes	151	149	8
i140701-003	-03	iso	2	yes	143	138	8
i140701-004	-05	rest	-	yes	147	147	7
i140702-001	-09	2-inst	1	yes	145	141	22
i140702-002	-04	iso	1	yes	142	142	15
i140702-003	-04	iso	2	yes	147	146	14
i140703-001	-03	2-inst	1	yes	156	150	19
i140703-002	-03	iso	1	yes	147	144	16
i140704-001	-04	2-inst	1	yes	136	134	11
i140704-002	-02	iso	1	yes	140	135	12
i140710-001	-08	iso	1	yes	136	126	33
i140718-001	-03	2-inst	1	yes	124	117	24
i140721-002	-04	2-inst	1	yes	114	109	28
i140725-002	-06	2-inst	1	yes	123	117	12
i140801-001	-03	iso	1	yes	112	109	13
i140917-002	-04	2-inst	1	yes	144	117	21
i141117-001	-03	2-inst	1	yes	92	89	16

Table A.11 – Overview of offline sorted datasets of second recording period of monkey N.