ACIT5900

MASTER THESIS

in

Applied Computer and Information Technology (ACIT)

May 2022

Biomedical Engineering

Creating a pipeline for functional connectivity analysis in fNIRS motor control studies

Sindre Lilleseth

Department of Mechanical, Electronic and Chemical Engineering

Faculty of Technology, Art and Design

Preface

The work behind this thesis has been the most interesting and challenging project of my life. I am tremendously enthusiastic about the opportunity to learn about one of the most complex systems in the known universe, the human brain, alongside being humble over the trust I have received from my supervisors during this period. With the objective of developing a tool for analysis of the human brain's functionality, I have done everything in my power to create the best possible result.

I want to use this opportunity to thank the people who have shown love and support and helped me during this exciting period of my life. I want to start with my partner, Gyda, who has supported me unconditionally throughout the five years leading up to this final project. I am also grateful for the support shown by my parents, Jenny and Erik, and my sister Emilie during these years. My friends' love and motivating words have also been essential in this period. My friend and fellow student, Håkon, has also been an important person during these years, and I am thankful for all the time we have spent learning, working, and having fun together. Additionally, I would like to thank Ph.D. Candidate Haroon Khan for all his important advice and helpful conversations.

Last but not least, I want to direct an extensive thank you to my supervisors Professor Olga Korostynska and Professor Peyman Mirtaheri. The way they have shared their knowledge and enthusiasm during these years has made me fall in love with science and the field of biomedical engineering.

Sindre Lilleseth

Oslo, Norway, May 2022

Abstract

Functional near-infrared spectroscopy (fNIRS) has shown to be a useful imaging technique for observing the human cortex. Its capabilities of imaging the hemodynamic variations in the cerebral cortex during different experimental paradigms enable studies that can contribute to unraveling the orchestration of the mind. With many questions regarding its workings still unanswered, different techniques are being employed in the analysis of fNIRS data to illuminate activation patterns and correlations between brain areas during different tasks. One technique that is gotten more attention during the last years, is functional connectivity analysis which aims to assess the temporal correlations between these areas. With the potential of assessing the cortical connectivity, this technique can contribute to analyzing people with different health challenges, both regarding mental and motor health.

However, with a lack of standards in pre-processing and processing techniques the analysis of such studies is known to be time-consuming and challenging. The different processing software and toolboxes are taking different approaches to perform these steps and with little insight into how the data is processed during these steps, a consistent and transparent pipeline is desired to enable reliable results in fNIRS motor control studies. This has led to the aim of mapping the most promising techniques used in pre-processing and processing of motor control fNIRS studies and implementing these techniques in an automatic processing pipeline. Thus, the objective of this study has been to develop such a pipeline to facilitate faster and more consistent functional connectivity analysis.

The developed pipeline is thoroughly described in this thesis, and its workings are presented using data from an ongoing motor control study.

Table of contents

List of figures

List of tables

List of abbreviations

BOLD	-	Blood Oxygen Level Dependent
DPI	-	Dot Per Inch
DPF	-	Differential Pathlength Factor
EC	-	Effective Connectivity
EEG	-	Electroencephalography
FC	-	Functional Connectivity
fMRI	-	Functional Magnetic Resonance
fNIRS	-	Functional Near-Infrared Spectroscopy
GLM	-	General Linear Model
GUI	-	Graphical User Interface
Hb	-	Deoxygenated Hemoglobin
Hb02	-	Oxygenated Hemoglobin
HCP	-	Human Connectome Project
mBLL	-	Modified Beer-Lambert Law
MEG	-	Magnetoencephalography
PET	-	Positron Emission Tomography
PFC	-	Prefrontal Cortex
PMA	-	Pre-Supplementary Motor Area
PMC	-	Premotor Cortex
PMS	-	Premotor and Supplementary Cortex
PPF	-	Partial Pathlength Factor
PSD	-	Power Spectral Density
PVF	-	Partial Volume Factor
QCoD	-	Quartile Coefficient of Dispersion
ROI	-	Region of Interest
SGWA	-	Supramarginal Gyrus of Wernicke’s Area
SMA	-	Supplementary Motor Area
SMC	-	Sensorimotor Cortex

1 Introduction

The famous scientist Michio Kaku (n.d.) once said "*The human brain has 100 billion neurons, each neuron connected to 10 thousand other neurons. Sitting on your shoulders is the most complicated object in the known universe*". With the desire to unravel the workings of the human mind, researchers worldwide are constantly investigating how the human brain operates. As technological innovation continues to push science forward, the mysteries of the brain are being dissected slowly. By assessing the correlations in activation between different brain areas, the orchestration of the brain is being investigated.

One burgeoning technique in neuroimaging is functional near-infrared spectroscopy (fNIRS) which can measure the changes in hemodynamic variations in the cerebral cortex (Irani et al., 2007). Since its implementation in 1977, fNIRS has been a non-invasive neuroimaging technique able to assess the activity in outer regions of the brain as it measures the hemodynamical variations in the region of interest (ROI) (Yucel et al., 2021). Hemodynamic variations depend on the neuronal metabolism that occurs in activated regions dependent on glucose and oxygen. As oxygenated and deoxygenated hemoglobin is the main absorption component of near-infrared light, the fNIRS imaging technique enables measurements of hemodynamic cortical responses (Pinti et al., 2020).

The mobility of fNIRS brain imaging technology enables observations of the brain in different states as the participants can perform different tasks regarding psychological behavior and motor control (Dans et al., 2021). Its non-invasive, unharmful, and modest influence on the participants has proven to be useful in a variety of studies (Yucel et al., 2021). Motor control studies aim to map how the brain function and have previously been challenging to conduct. Therefore, the development of portable imaging techniques enables studies to be conducted in environments that are more closely related to real-life scenarios, which can lead to the acquisition of new and valuable information (Dans et al., 2021).

The process of fNIRS studies is separated into several necessary stages starting with the data acquisition and ending with result interpretations. The raw data from the acquisition using the fNIRS cap goes through steps to convert the raw signals to plots and key coefficients for result interpretation. This range of steps is key in processing the signals and often follows a given set of processes to run through. However, with constant development within the field, there are no current standard procedures for pre-processing and processing the fNIRS data.

This can lead to inconsistent representation and analysis of fNIRS data and unnecessary processing time (Yucel et al., 2021). Together with an absence of transparent and standardized study protocols in fNIRS studies, challenges related to comparability and interpretation of studies can lead to limitations in the development of the field (Herold et al., 2017).

Given the lack of a standardized and effective pipeline for fNIRS studies, this thesis aims to map out how researchers are processing the fNIRS data and propose a pipeline to be used by both researchers and students in the future fNIRS motor control studies. With a primary motivation of facilitating effective, reliable, and transparent fNIRS functional connectivity analysis for motor control studies the following research questions are raised:

1. What is the state-of-the-art procedure for analyzing motor control fNIRS data?
2. What features are most important to include in a pipeline for functional connectivity analysis of fNIRS data from motor control studies?

These questions provide the basis for the objective of this thesis which has been to develop a processing pipeline for fNIRS motor control studies to enable functional connectivity analysis. To present the workings of the pipeline, data obtained in a recent motor control study by OsloMet is processed and reviewed.

The structure of the thesis is based on the IMRaD structure where the following chapter includes the background together with the relevant literature review behind this work. Then, the methodology chapter that outlines how the development of the pipeline is conducted alongside a description of its features. Then the result of this work is presented, followed by discussions and a conclusive summary of the thesis. Lastly, recommendations for future directions are provided.

2 Background and literature review

In this chapter, the foundational literature behind this work is presented to illustrate the field’s status and progress and what challenges the researchers face. Moreover, to provide a common understanding of critical aspects relevant to this thesis, the challenges related to fNIRS pipelines and study protocols are presented together with some of the main ideas and concepts related to neuroimaging and motor control studies.

2.1 Functional neuroimaging and cortical activity

The investigation of the human brain’s functionality is constantly moving forward. To explore how the brain operates neuroimaging techniques are being employed to map the activation areas and their connection to other areas. Some of the most important techniques that have contributed to the field are functional magnetic resonance (fMRI), positron emission tomography (PET), electroencephalography (EEG), and magnetoencephalography (MEG) (Crosson et al., 2010). These techniques have enabled modeling of the spatiotemporal brain activation and brought the study of brain network organization, known as connectomics, further during the last decades (Bullmore et al., 2016).

Today the field of connectomics is working towards understanding the structural and functional connectivity of the brain. The structural approach aims to map the physical neuronal connections between different brain regions, while the functional approach explores the operational relationship between different brain regions in a spatiotemporal relation (Wong et al., 2020). Essential in the development of structural and functional connectivity is the implementation of graph theory to view the brain regions as nodes and edges and thereby generate a network representing the connectivity of the brain (Wang et al., 2015). The brain analysis-related literature found in the work on this thesis shows that a significant number of researchers focus on this approach for mapping connectivity.

The current paradigm related to cortical behavior within neuroscience and brain anatomy includes the parcellation of the cerebral cortex where the different areas are given an approximate location in the brain (Bullmore et al., 2016). One of the most recognized parcellations in the field is the mapping conducted by K. Brodmann in 1909 who divided the cortex into 48 areas based on the different cell types found in each area (Zilles, 2018). These areas have some variations between individuals that hinder a general parcellation (Van

Essen, 2013). Moreover, limitations in the spatial and temporal resolution, and the usability of the imaging techniques, have previously limited the interpretation of the collected data (Sporns et al., 2005). The mapping of the different areas is ongoing research, but scientists are getting closer to an understanding by using different approaches for parcellation. By combining multimodal imaging techniques, neuroanatomical approaches and machine learning algorithms for classification, a comprehensive study related to the Human Connectome Project (HCP) conducted by Glasser et al. (2016) mapped the cortical areas of 210 healthy young adults. As shown in Figure 1, they reported an even more detailed separation of the areas, and their study stands as one of the most important studies in this field in recent times (Elam et al., 2021). However, the connectivity of the brain is still an ongoing field of research with a lot of functionality that is still unknown.

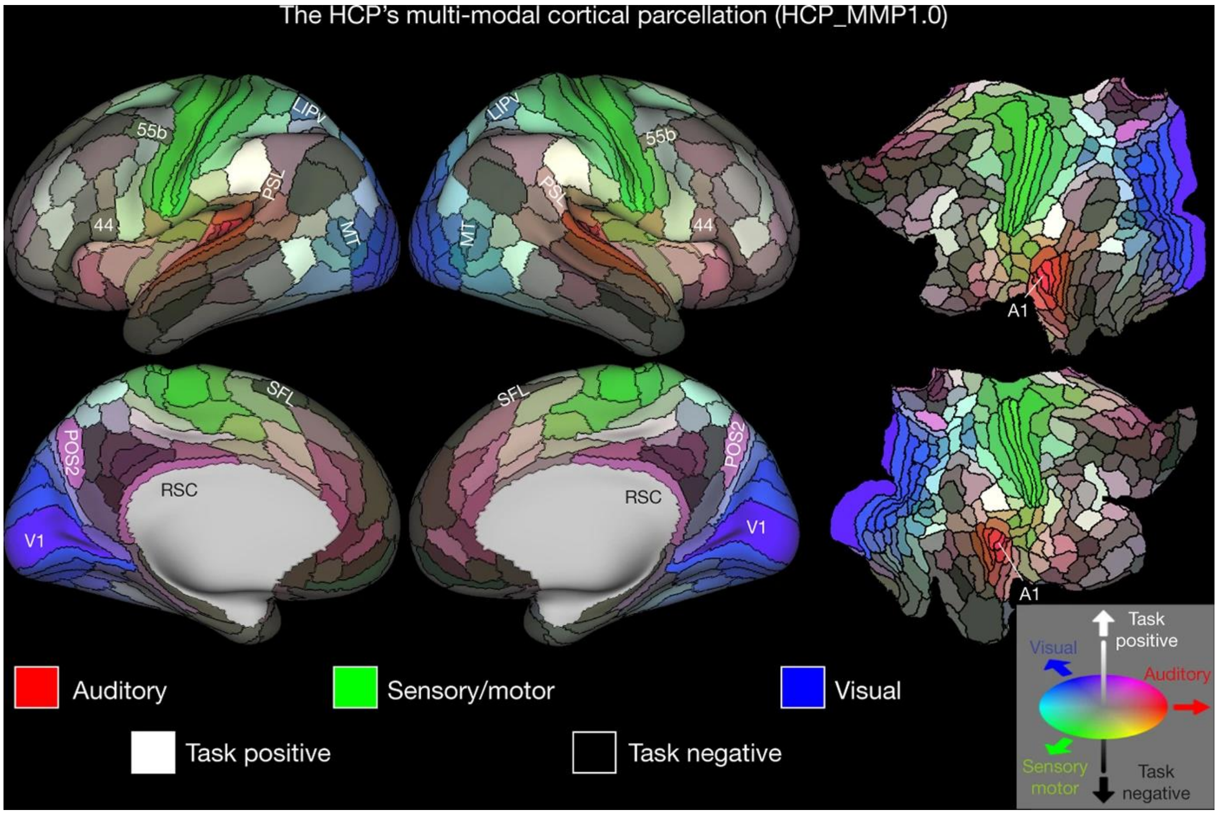


Figure 1: The cortical parcellation from Glasser et al. (2016). (https://www.nature.com/articles/nature18933/figures/3).

One of the most utilized techniques within this field is fMRI which can measure the blood oxygen level-dependent (BOLD) signals that are a direct measure of the oxygenation within a brain area. The signal depends on the paramagnetic properties of the deoxygenated hemoglobin and gives information regarding the activation of different areas in the brain based on its neuronal metabolism and has brought the field forward during the last decades

(Huppert et al., 2006). However, as fNIRS enables the measurement of both the oxygenated and the deoxygenated hemoglobin, the hemodynamic state in a specific cortical area can be analyzed using an additional parameter compared to fMRI (von Luhmann et al., 2020). The differences between the signal obtained by fNIRS and the BOLD signal from fMRI are illustrated in Figure 2.

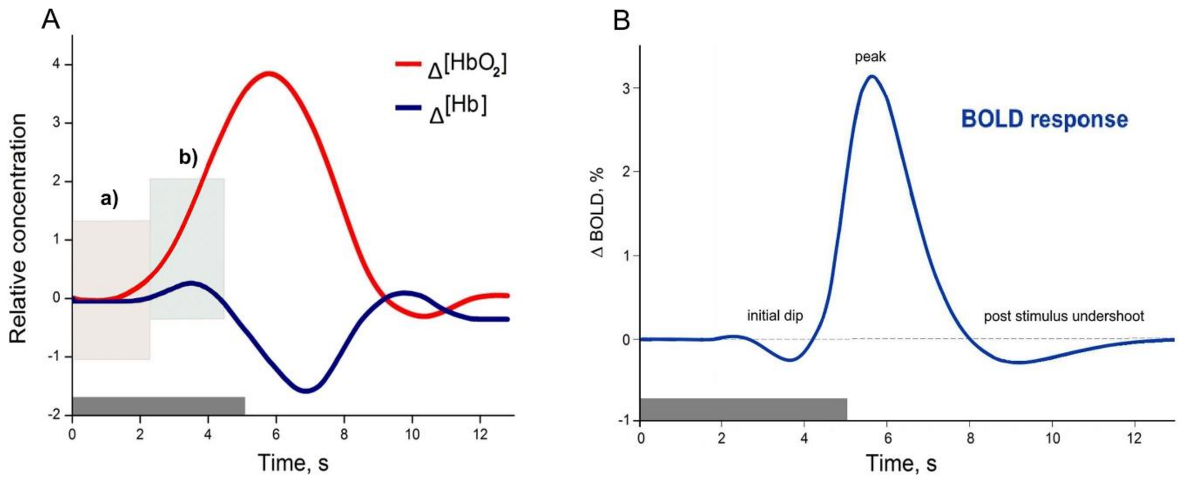


Figure 2: Illustration of (A) the hemodynamic response measured in fNIRS and (B) the BOLD response measured in fMRI. By Cinciute (2019).

Additionally, the development of modern fNIRS equipment enables active monitoring of brain activation in scenarios where other techniques are impossible to employ as they have limitations regarding either wearability or spatiotemporal resolution (Pinti et al., 2020). This makes fNIRS a promising tool for functional imaging of cortical activity.

2.2 fNIRS data processing

The optical nature of fNIRS imaging ensures a non-ionizing, non-invasive, and practical detection of brain activity. By transmitting light in the near-infrared area through the tissue of the skull, the photon path goes through several layers of different tissue (Pinti et al., 2020). These layers are illustrated in Figure 33 and influence the fNIRS signal as the photons are scattered and absorbed by the chromophores in the tissue outside the region of interest (ROI). Additionally, the figure also illustrates the differences between short separation channels and long separation channels.

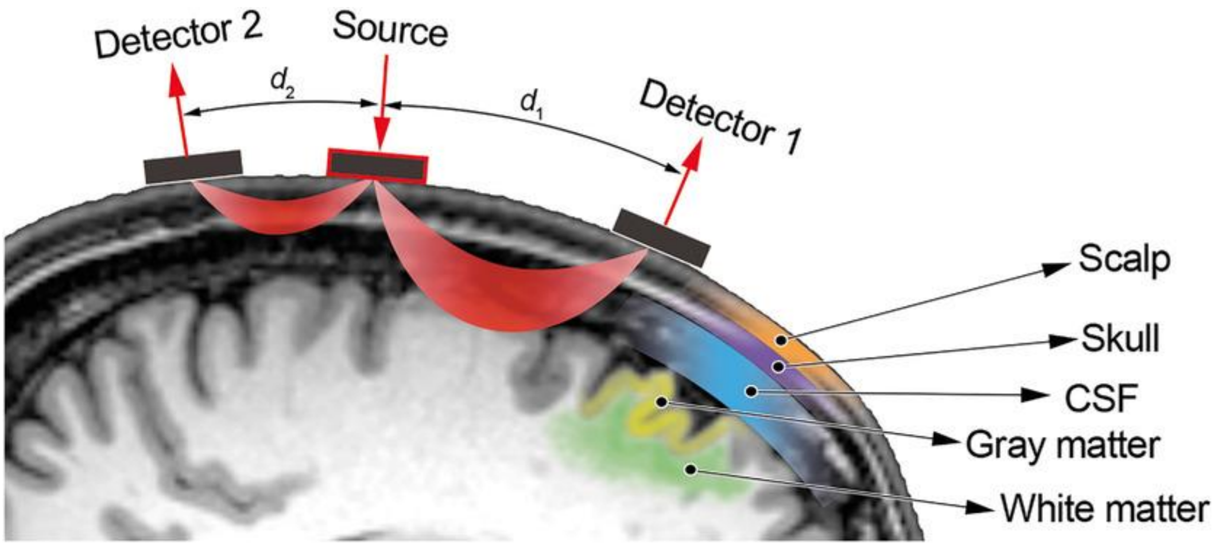


Figure 3: Illustration of the photon path in fNIRS. By Pinti et al. (2018). (https://nyaspubs.onlinelibrary.wiley.com/doi/full/10.1111/nyas.13948)

As the most prominent chromophores in the near-infrared area (650-950 nm (Strangman et al., 2003)) are the oxygenated and deoxygenated hemoglobin, these layers influence the optical signal by the variations in blood flow in the areas outside of the cortex (Dans et al., 2021). Figure 44 illustrates the optical absorption spectra of oxy- and deoxygenated hemoglobin, which is the main enabler for fNIRS measurements. The combination of these factors leads to the need for the processing of the signals.

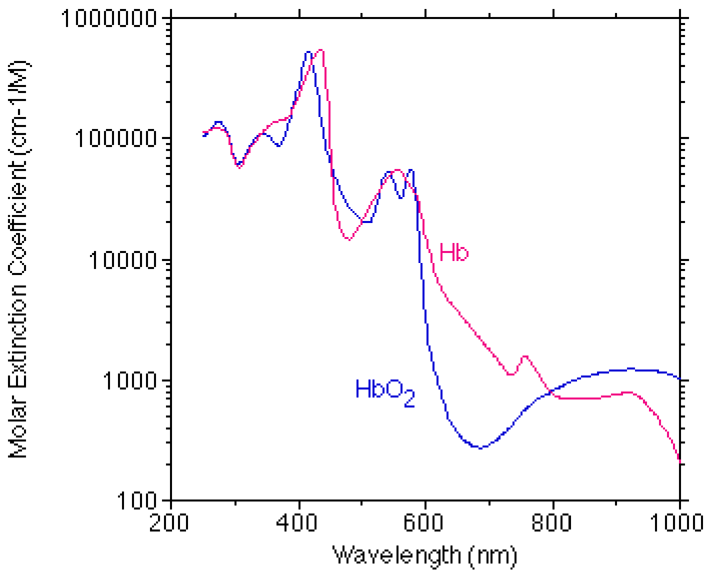


Figure 4: The optical absorption spectra of hemoglobin. By S. Prahl. (https://omlc.org/spectra/hemoglobin/).

This processing is divided into several steps that are necessary to obtain interpretable information regarding the hemodynamics in the tissue. These steps are presented more thoroughly in chapter 3.2 of this thesis, but a brief explanation is also provided in this section. The first step in this data processing is to convert the signal from the raw voltages received by the optodes in the acquisition equipment. The raw signal, which is a direct representation of the voltages produced by the optical detectors after receiving the near-infrared light, is converted to optical density. Then, the optical densities are converted to hemoglobin concentrations using the modified Beer-Lambert Law (mBLL). This step employs a factor, either the differential pathlength factor (DPF) or the partial pathlength factor (PPF), to compensate for the increase in the photon's pathlength within the tissues as it is scattered. The difference between these two factors is further explained in chapter 3.3.2. Other steps that are often employed during pre-processing of the signals are the application of frequency filters, removing channels, wavelet filtering, correction of motion artifacts, and other techniques for deriving the signal of interest (Dans et al., 2021). These steps are presented in Figure 5.

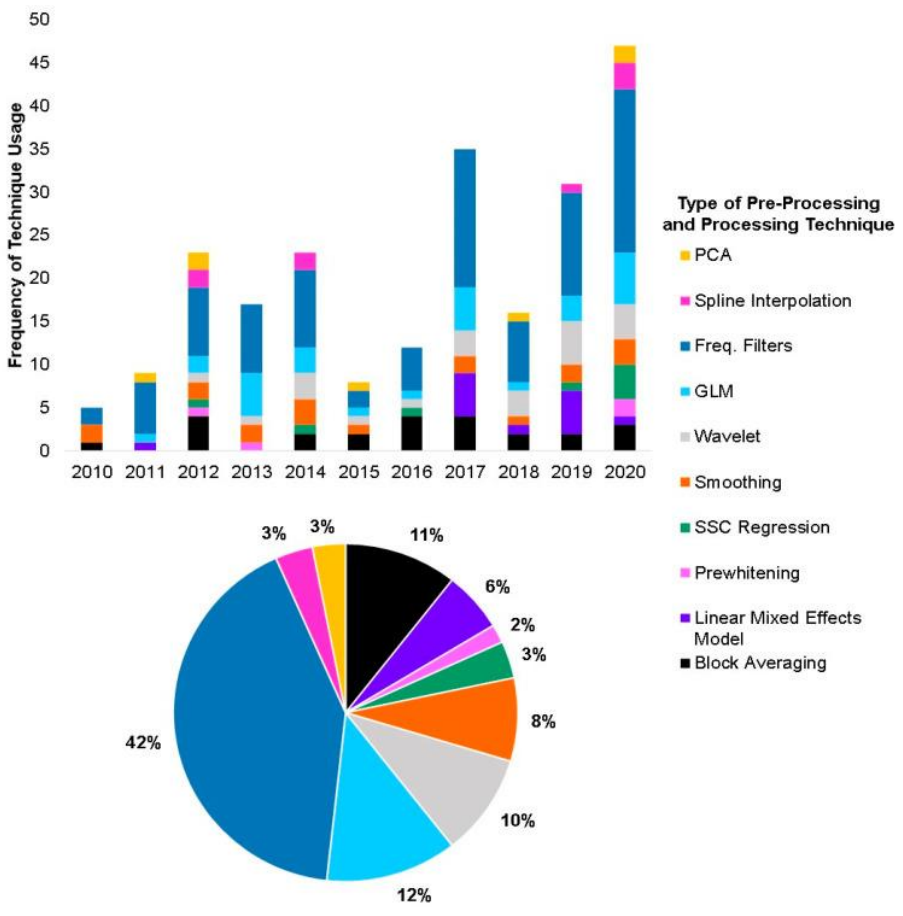


Figure 5: Types of pre-processing and processing techniques used in fNIRS studies. By Dans et al. (2021).

When the pre-processing of the data is conducted, different processing algorithms can be applied to derive statistical information from the dataset. During the last decade the three most used processing techniques are the general linear model (GLM), block averaging, and linear mixed models (Dans et al., 2021). These techniques are used based on the related aims of the studies and provide different data. GLM is a much-used technique in the fMRI field and has been employed in the processing of fNIRS data because of the similarities between the BOLD signals and the hemodynamic response signal (Monti, 2011). Block averaging and linear mixed models are also popular methods to use, but the GLM is known for its wide use.

However, the most used techniques are known to not necessarily best (Dans et al., 2021). With a vast number of possible techniques to employ, and in a continuously evolving research environment, they are not to be depreciated. One of these techniques is functional connectivity (Fantini et al., 2018).

2.3 Functional connectivity

In mapping the connectivity of the human brain, researchers are using different statistical techniques to measure the dependencies between cortical areas (Bullmore et al., 2016). By assessing the temporal correlations and causality, and spectral coherence between spatially distant neurophysiological events, functional connectivity is a concept that is frequently implemented in brain connectivity studies (Friston, 2011). As functional connectivity looks at the similarities in the activation and behavior between the different areas in the brain, interesting information can be collected regarding cortical functionality.

Functional connectivity can be measured in the time and frequency domain (NIRx Medical Technologies, 2019). The techniques used for time-domain analysis are related to the temporal correlation between time-series signals, while frequency-based techniques evaluate the coherence between frequencies between signals. Thus, the connectivity can be measured using different statistical measures for the relationship between the regions of interest and is a flexible tool for brain connectivity analysis (Eickhoff & Müller, 2015).

One of the most used statistical measures for temporal correlation used in functional connectivity is the Pearson correlation coefficient (also known as Pearson's R) which is a measure of the linear correlation between time series signals (Bullmore et al., 2016). This coefficient describes the correlation between the signals with a number ranging from -1 to 1, where 1 means an identical behavior between the time series data and -1 means a negative correlation. An R-value close to zero indicates no linear correlation (Bullmore et al., 2016). Other measures of temporal correlation are cross-correlation and Spearman correlation (Xu et al., 2015).

Other approaches used in connectivity analysis are Coherence and Granger causality which employ different techniques for detecting similarities in signals. Coherence is related to analyzing the frequency spectrum of the signals and can provide information regarding the functional connectivity within the frequency domain (Bowyer, 2016). This technique has proven useful in spectral analysis, but is known for being prone to changes in signal amplitude and can therefore have reduced precision in motion control studies (Bullmore et al., 2016). Granger causality is related to the concept of Effective Connectivity (EC) and goes one step further in the connectivity analysis as it looks for causality within the temporal brain data (Bowyer, 2016). This is a promising technique for connectivity analysis as it describes the directionality between brain areas. However, the technique is still being investigated for use in fNIRS studies as it has proven challenging to describe causality in connectivity analysis (Anwar et al., 2016).

Even though Pearson's R-value is the most used measure for FC, there is no current gold standard of which technique to apply for functional connectivity analysis (Blanco et al., 2018). Researchers are encouraged to choose the best method for their experimental design (Yucel et al., 2021). However, one of the most promising techniques for conducting FC processing in fNIRS studies is the use of autoregressive models based on the pre-whitening of the signals (Blanco et al., 2018).

The use of functional connectivity in fNIRS-based studies has previously been primarily used in resting-state experiments (Xu et al., 2015). With little implementation of the technique in motor control studies, guidelines related to its use is missing within the field. However, with an increasing number of published articles regarding functional connectivity in fNIRS studies each year, the field is moving forward (Fantini et al., 2018).

2.4 Frequencies in cortical hemodynamics

The data from the fNIRS measurements consists of a range of low-frequency bands connected to different physiological states. These oscillations erupt from the hemodynamic variations measured by the optodes and exist in the range from 0.0095 Hz to 2 Hz (Yücel et al., 2016). Within this frequency range, several areas are separated into subareas connected to their associated physiological states. Figure 6 illustrates how these subareas are divided and how much influence they have on fNIRS signals.

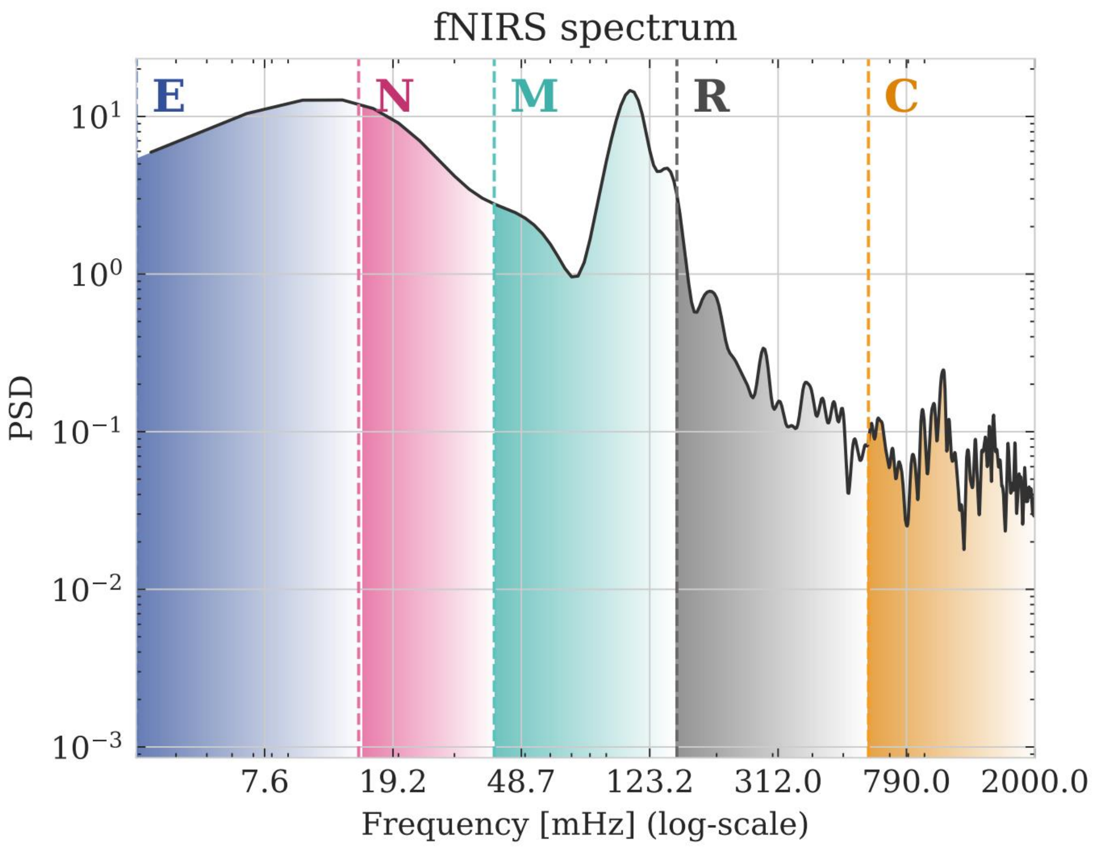


Figure 6: Sample power spectral density (PSD) in fNIRS signals. By Pinto-Orellana (2022).

In the area of 3-20 mHz, endothelial waves (E) are a known influence on the signal, while the area of 20-50 mHz is connected to neurogenetic components (N). Myogenic waves (M) are observed in the 50-150 mHz area (Pinto-Orellana, 2022). In the 100 mHz area blood pressure-related oscillations, known as Mayer waves, are also known to influence the optical signal. Signal fluctuations related to respiration (R) are known to be in the area of 0.15-0.4 Hz, and physiological noise caused by cardia noise (C) is in the area of 1-1.5 Hz (Klein & Kranczioch, 2019). These different frequency areas are necessary to consider when analyzing fNIRS signals as they influence the data that is analyzed. This concept is relevant for the filtering of the signal.

2.5 Motor control

One of the clear advantages that fNIRS technology adds to the field of brain imaging, besides its non-invasive and non-ionizing nature, is its portability (Yücel et al., 2017). This enables brain imaging during tasks that require both spatial relocation and thereby exploration of cortical activity during body movement. The study of neuronal behavior during movement is known as motor control and is defined as the “area of science exploring natural laws that define how the nervous system interacts with other body parts and the environment to produce purposeful, coordinated movements” (Latash, 2012).

Motor control studies contribute to the revelation of how the nervous system communicates these coordinated movements between the brain and the limbs. As the regions related to motor functions are located near the skull, fNIRS imaging is considered a suitable technology for measuring their activity.

2.6 Postural balance and gait studies

One of the complex movements humans perform is the postural balance required during gait (Herold et al., 2017). This task requires the cooperation between several joints and muscles which are controlled using different strategies. These strategies are often divided into three main categories: motor, sensory, and cognitive strategies (Shumway-Cook & Woollacott, 2017, pp. 277-306). All these strategies contribute to the human gait and by analyzing the cortical activity in different scenarios, researchers aim to map the spatiotemporal activation of the brain.

Previous motor control studies have aimed to improve rehabilitation strategies by evaluating brain activity from walking tasks and tasks that require postural balance. Some of these studies have shown that activations in several areas occur during walking: the prefrontal cortex (PFC), pre-supplementary motor area (PMA), premotor cortex (PMC), supplementary motor area (SMA), and sensorimotor cortex (SMC) (Herold et al., 2017). It indicates that walking is not only constricted to the activation of one single cortical area. In the analysis of postural balance, studies have shown that PFC and SMA are the most prominent areas (Herold et al., 2017). One study which investigated the functional connectivity using EEG during walking measured a decrease in FC during walking compared to standing (Lau et al., 2014).

During gait assessment, differences in the cortical activity have also been found related to factors such as age, level of fitness and health, and the complexity of the tasks (Hamacher et al., 2015). Such factors are also known to impact the different parameters used in the processing of fNIRS, such as the PPF and DPF (Yucel et al., 2021). Even though studies have revealed some of the activation and functional connectivity during gait and postural balance, additional research is needed to map the complete orchestration of the brain during balance and gait.

2.7 State-of-the-art fNIRS software

The field of fNIRS is in constant movement and the researchers are pushing the field forward each year. The growing number of published studies that use fNIRS as the neuroimaging technique generate new insight into how the human brain works. The lack of physical constraints and its non-invasive nature enables brain activation studies to be performed in settings that previously not have been available (Pinti et al., 2018). Data acquisition and processing are getting better with the constant improvement in fNIRS technology. However, like the early development of the fMRI field, fNIRS researchers are lacking standards in data processing and analysis (Yucel et al., 2021).

2.7.1 Software and toolboxes

A large number of different optical imaging devices and software results in different processing pipelines used by researchers. This leads to inconsistency in data interpretation and reporting (Bonilauri et al., 2021). As presented in Table 1, there are many different tools available for the analysis of fNIRS data. This gives a vast number of different scripts and functions for handling the fNIRS data, which often lead to different processing and interpretation methods (Dans et al., 2021).

Table 1: Overview of well-known software and toolboxes in fNIRS analysis.

Software	**Category**	**Interface**	**Platform**
fNIRSOFT	fNIRS analysis	Stand-alone	Stand-alone
fOSA-SPM	fNIRS analysis	GUI[1]	MATLAB-based
FC-NIRS (Xu et al., 2015)	Functional Connectivity Analysis	GUI	MATLAB-based
fOLD (Zimeo Morais et al., 2018)	Optode localization	Toolbox and GUI	MATLAB-based

[1] GUI – Graphical User Interface

HOMER3 (Huppert et al., 2009)	fNIRS analysis	Toolbox and GUI	MATLAB-based
HOMER2 (Huppert et al., 2009)	fNIRS analysis	Toolbox and GUI	MATLAB-based
ICNNA (Orihuela-Espina et al., 2017)	fNIRS analysis	GUI	MATLAB-based
MNE-NIRS (Gramfort et al., 2013)	fNIRS analysis	Toolbox	Python-based
NeuroDOT (Muccigrosso & Eggebrecht, 2018)	fNIRS analysis	Toolbox	MATLAB-based
NIRFAST (Dehghani et al., 2009)	fNIRS analysis	Toolbox	MATLAB-based
nirsLAB (Xu et al., 2014)	fNIRS analysis	GUI	MATLAB-based
NIRSite	Optode localization	GUI	Stand-alone
NIRS-SPM (Ye et al., 2009)	fNIRS analysis	GUI	MATLAB-based
NIRSTORM (Tadel et al., 2019)	fNIRS analysis	GUI	MATLAB-based
NIRS Brain AnalyzIR (Santosa et al., 2018)	fNIRS analysis	Toolbox	MATLAB-based
Open PoTATo (Sutoko et al., 2016)	fNIRS analysis	GUI	MATLAB-based
Turbo Satori	fNIRS analysis	GUI	Stand-alone
LIONirs (Tremblay et al., 2022)	fNIRS analysis	Toolbox	MATLAB-based

Another aspect within the field that is derived from Table 1 is the large number of graphical user interfaces and toolboxes that is MATLAB-based. With 14 of the 18 most used software utilizing MATLAB as the processing platform, the MATLAB engine seems to have an important role in the development within the field.

For functional connectivity analysis in fNIRS studies, only one of the software mentioned in Table 1 is designed with that functionality. However, some MATLAB toolboxes such as the NIRS Brain AnalyzIR toolbox by Santosa et al. (2018) have functions designed for such analysis.

2.7.2 File formats

The fNIRS data that is obtained by the measurement equipment is stored in different file formats. Different formats are available, and it is the standards employed by the manufacturer that decides which one is available for the specific device (NIRx Medical Technologies, 2020). Two of the most used formats of today are the .nirs and .snirf formats. They both have most of the same features, but the .snirf format is the newest format developed by the fNIRS community which working on implementing it as a standard to generate a common format for sharing data (fNIRS.org, 2022). This enables both formats to be employed when generating a pipeline for fNIRS processing.

2.8 Objective of the thesis

With the lack of state-of-the-art processing pipelines within the field of fNIRS and no golden standard for functional connectivity analysis of motor control studies, the objective of this thesis has been to develop a processing pipeline for functional connectivity analysis for fNIRS motor control studies. Motivated by the research collaboration between Oslo Metropolitan University and Gaitline AS, this thesis aims to facilitate effective and reliable pre-processing and processing of fNIRS data by implementing the most promising techniques found in the literature. The pipeline will hopefully relieve researchers from the time-consuming task of processing the data and let them perform what they do best, analyze and understand. Ultimately, the pipeline will enable important studies that can make life better for people in need and contribute to the unraveling of the workings of the human brain.

3 Methodology

This section presents the development of the pipeline and its features, followed by the experimental design of the previous motor control study conducted by Gaitline and Oslo Metropolitan University. Firstly, the rationale behind the design of the pipeline is presented alongside information regarding its dependencies and functionality. A concise description of each step employed in the pipeline is then provided to ensure understanding and transparency of its workings. Lastly, the experimental design of the motor control study is briefly presented as the fNIRS data collected in that study is used to illustrate the workings of the pipeline.

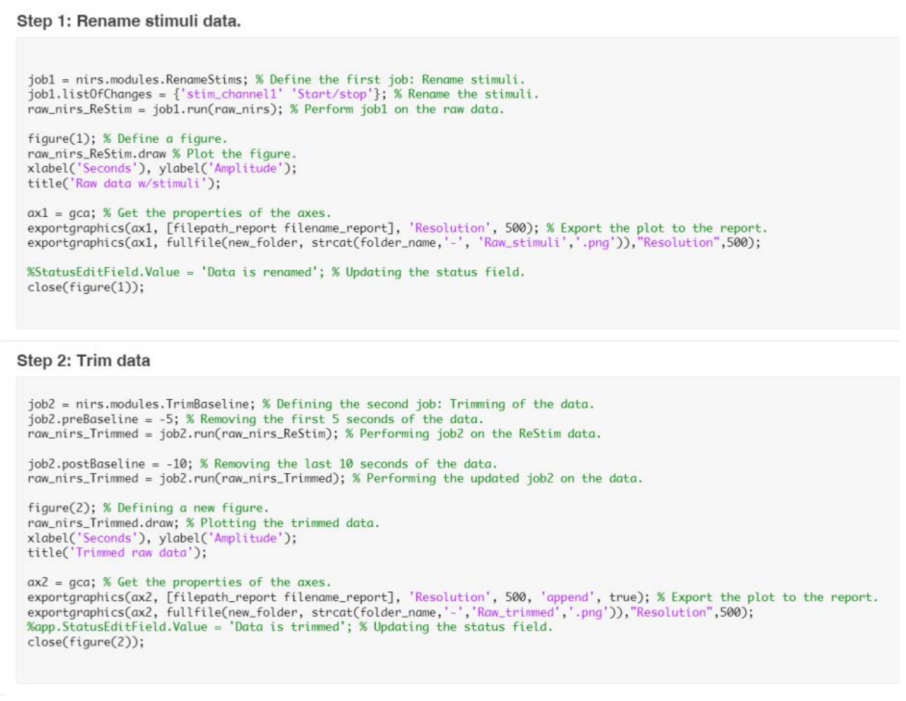
3.1 Development of the pipeline

As the primary goal of this thesis has been to develop a pipeline for pre-processing and processing of fNIRS data, state-of-the-art literature has been reviewed to map out the most promising techniques employed by researchers within the field. The findings from this review show that the field is under constant exploration and has no prominent golden standard for pre-processing and processing of fNIRS data. Especially not related to motor control studies.

3.1.1 MATLAB as a platform

With a vast number of available platforms for fNIRS analysis (see table 1), no software is known to be the golden standard for data processing of fNIRS signals. Some of the platforms are developed by the industry and others are developed within academic institutions. The stand-alone software from the industry often gives little transparency as the user is only given the GUIs and the user manual, without little insight into the source code. This removes important transparency within processing steps and can lead to the wrong interpretation of results as the user cannot interpret the actual workings of the software. As presented in table 1, the platforms developed by the academic institutions are often developed in the MATLAB environment and are provided either as MATLAB-based GUIs or toolboxes. This enables the user to utilize the GUI as a basic interface when processing the data or using the toolbox to create their own processing pipeline.

However, these GUIs and toolboxes are known to be challenging to implement and utilize as there are compatibility challenges related to fNIRS acquisition systems, data formats, and MATLAB versions. This has led to the aim of creating a user-friendly processing pipeline that utilizes the processing power of the MATLAB software together with an interface that is easily interpreted and utilized. To ensure a transparent program that also enables processing efficiency, the program developed in this work is both providing a stepwise live script in the MATLAB R2021b environment and a simple GUI in the MATLAB R2021b app designer environment. Figures 7 and 8 present the two interfaces.



Step 1: Rename stimuli data.

```
job1 = nirs.modules.RenameStims; % Define the first job: Rename stimuli.
job1.listOfChanges = {'stim_channel1' 'Start/stop'}; % Rename the stimuli.
raw_nirs_ReStim = job1.run(raw_nirs); % Perform job1 on the raw data.

figure(1); % Define a figure.
raw_nirs_ReStim.draw % Plot the figure.
xlabel('Seconds'), ylabel('Amplitude');
title('Raw data w/stimuli');

ax1 = gca; % Get the properties of the axes.
exportgraphics(ax1, [filepath_report filename_report], 'Resolution', 500); % Export the plot to the report.
exportgraphics(ax1, fullfile(new_folder, strcat(folder_name,'-', 'Raw_stimuli','.png')),"Resolution",500);

%StatusEditField.Value = 'Data is renamed'; % Updating the status field.
close(figure(1));
```

Step 2: Trim data

```
job2 = nirs.modules.TrimBaseline; % Defining the second job: Trimming of the data.
job2.preBaseline = -5; % Removing the first 5 seconds of the data.
raw_nirs_Trimmed = job2.run(raw_nirs_ReStim); % Performing job2 on the ReStim data.

job2.postBaseline = -10; % Removing the last 10 seconds of the data.
raw_nirs_Trimmed = job2.run(raw_nirs_Trimmed); % Performing the updated job2 on the data.

figure(2); % Defining a new figure.
raw_nirs_Trimmed.draw; % Plotting the trimmed data.
xlabel('Seconds'), ylabel('Amplitude');
title('Trimmed raw data');

ax2 = gca; % Get the properties of the axes.
exportgraphics(ax2, [filepath_report filename_report], 'Resolution', 500, 'append', true); % Export the plot to the report.
exportgraphics(ax2, fullfile(new_folder, strcat(folder_name,'-','Raw_trimmed','.png')),"Resolution",500);
%app.StatusEditField.Value = 'Data is trimmed'; % Updating the status field.
close(figure(2));
```

Figure 7: An excerpt of the live script interface of the pipeline.

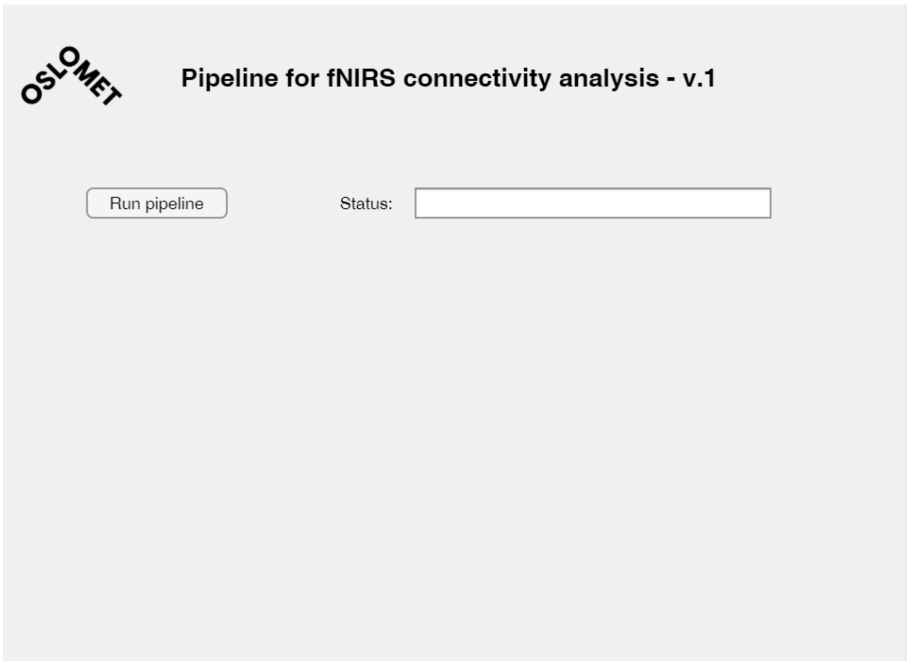


Figure 8: An illustration of the GUI of the pipeline.

The live script is developed to allow the user to run the pipeline stepwise as each preprocessing and processing step is separated into code blocks that can be run separately. Alternatively, the script can be run from start to finish by pressing the “run” button in the MATLAB environment. This also ensures the possibility of using the code as a framework for further changes to the pipeline and to have the flexibility that other GUIs do not provide. However, by operating in the same window as the script, the user is prone to interfere with the script in unwanted manners and make changes that might affect its functionality. Therefore, a simple GUI is provided to remove the possibility of interference and to simplify the user interaction.

3.1.2 The Brain AnalyzIR Toolbox

When developing a pipeline for fNIRS analysis, there are several aspects to consider and challenges to overcome. With several available fNIRS systems and toolboxes for data acquisition and processing, the fNIRS community has many contributors who work towards improvement of the field (Yucel et al., 2021). Different standards, such as the ‘.snirf’ file format, are also being suggested to facilitate consistency in fNIRS studies and cooperation across institutions. However, from experience, these different factors can make it challenging to perform the processing and to use the necessary functions when working with the data. Incompatibility between MATLAB versions and the toolboxes is often creating challenges during processing and can lead to more work for the researcher and a longer time between measurement and research publishment.

As the current focus within the fNIRS community is to adapt the .snirf file format, I found it necessary to use a toolbox that both were compatible with this format and also had the necessary functions needed for the processing pipeline. During the preparations for the pipeline development in this project, a review of the functionality of the different toolboxes revealed that the toolbox provided with the HomER3 package by Huppert et al. (2009) had the necessary functions and compatibility needed. However, during development, limitations regarding its functionality together with compatibility issues regarding the .snirf format occurred, and I found it necessary to rebuild the pipeline using another toolbox. The Brain AnalyzIR Toolbox by Santosa et al. (2018) worked well together with the .nirs format and is the implemented toolbox. The toolbox is developed by recognized researchers within the field and has the functions necessary for building an fNIRS pipeline for motor control studies.

Alongside the need for the toolbox for fNIRS processing, the standard MATLAB toolbox called the Signal Processing Toolbox is required for using the pipeline.

3.2 Features of the pipeline

The pipeline is developed with the aim of enabling fast and structured processing of fNIRS data obtained from motor control studies conducted by Oslo Metropolitan University and Gaitline AS. To facilitate such studies, pre-processing and processing steps that have shown promising results in previous motor control studies are employed to streamline the process.

In this section, the chronological process of the pipeline is described focusing on the functionality of the employed algorithms and the mathematics behind the most important ones. This is provided to ensure transparency, work as a guideline for users, and enable relevant referencing if using the pipeline for future studies. As presented in Figure 9, the pipeline consists of 15 steps that include loading, pre-processing, and processing of the data, parallel to generate a report with the outcomes of the pipeline. These steps are identical for both the live script and the GUI. The only difference between them is the interface used by the user. The design of the pipeline aims to adhere to the recommendations for fNIRS studies presented in Yucel et al. (2021). A thorough description of the workings of the toolbox is presented in Santosa et al. (2018).

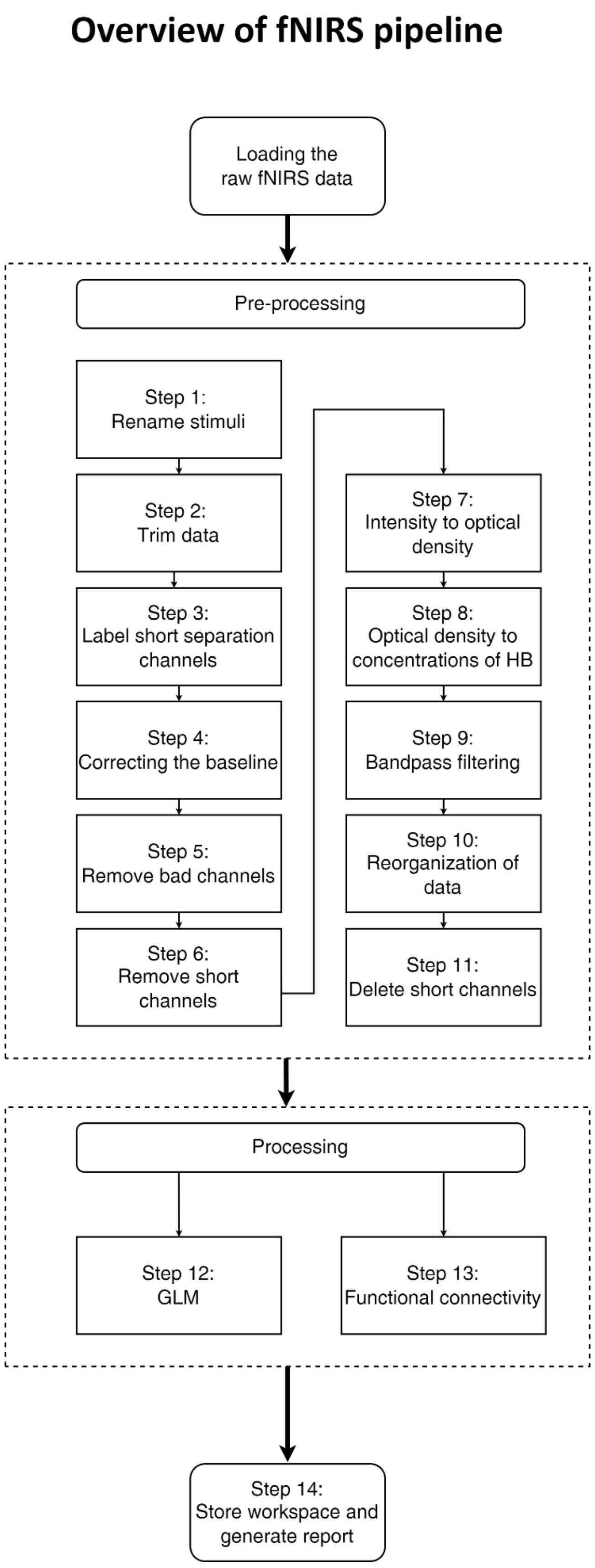

Overview of fNIRS pipeline
Loading the raw fNIRS data
Pre-processing
Step 1: Rename stimuli
Step 2: Trim data
Step 3: Label short separation channels
Step 4: Correcting the baseline
Step 5: Remove bad channels
Step 6: Remove short channels
Step 7: Intensity to optical density
Step 8: Optical density to concentrations of HB
Step 9: Bandpass filtering
Step 10: Reorganization of data
Step 11: Delete short channels
Processing
Step 12: GLM
Step 13: Functional connectivity
Step 14: Store workspace and generate report

Figure 9: Overview of the pipeline.

3.2.1 Loading data and creating the report

The initialization of the pipeline is the only phase that requires user interaction. When running the script, the user is asked to choose three folders: 1. The folder containing the repository, 2. The folder containing the test data, and 3. The folder for storing the report. After this phase, the pipeline runs until the report is entirely generated. Three MATLAB files are also loaded as these provide the correct information related to the areas based on the probe setup.

To load the data, the nirs toolbox function named “nirs.io.loadDotNirs” is employed to enable the loading of the .nirs file format. This creates a data object with the “nirs.core.data” structure. The content of the data object consists of the raw data, probe information, accelerometer and gyroscope data, stimuli information, and the sampling frequency. A further explanation of the data structure is presented in Santosa et al. (2018).

For each step of the pipeline, a report is generated consisting of the plots created after each processing step. This feature is provided to enable the user to visually inspect the outcomes of each step as well as having a document that includes all the relevant plots. Each plot is also stored as single files within a folder that is placed in the same location as the report. The plots are stored as .png files with 500 dots per inch (DPI) resolution to give high resolution for implementation in scientific papers.

3.2.2 Pre-processing

Step 1 to 11 in the script consists of pre-processing steps that are implemented to enhance the signal quality and are based on the most promising techniques stated in the literature. These steps can all be found in motor control studies, but as there is no golden standard for the order of the steps the design of the pipeline is based on the functionality of the toolbox as well as the desired stepwise flow (Dans et al., 2021). The following paragraphs consist of a brief description of each step.

Step 1: Rename stimuli

Step 1 includes the renaming of the stimuli that are within the .nirs file. The automatic property of the function makes it flexible for different types of stimuli and automatically detects the stimuli information. As different studies can employ different stimuli, the only change the user needs to do is to insert the names of the stimuli in the script when running the script for the first time.

Step 2: Trimming data

Step 2 removes data points that are prone to noise from starting and stopping the experiment. The chosen default values are the first 5 seconds and the last 10 seconds of data recording. This is employed as it is common practice in motor control studies (Hocke et al., 2018), and as noise within these areas was visually observed in the raw data. However, this can arbitrarily be changed within the script as future experimental designs changes. The toolbox function used here is called “nirs.modules.TrimBaseline”.

Step 3: Short separation channel labeling

Step 3 provides labeling of the short separation channels to be able to distinguish them from the regular channels. This is done with the use of the “nirs.modules.LabelShortSeparation” function. Short channels are used to remove physiological noise from the data (Yücel et al., 2015).

Step 4: Baseline correction

Step 4 corrects the baseline shift within the raw data by removing the DC-shift provided by motion artifacts. The utilized toolbox function is named “nirs.modules.BaselineCorrection” and is further explained in Santosa et al. (2018).

Step 5: Bad channels rejection

A reported challenge in fNIRS motor control studies is to ensure good connections between the optodes and the scalp (Yucel et al., 2021). This step is commonly conducted through the time-consuming visual inspection of the spectral channel data. However, automatic spectral analysis can be performed by using different techniques. One of the most utilized techniques is calculating the power spectral density (PSD) and removing channels containing a high power of frequencies outside of the frequency area of interest (Aarabi & Huppert, 2016). Figure 10 and 11 illustrates the difference between an included channel and a rejected channel by plotting their respective PSD. The included channel contains a higher spectral power in the lower frequencies while the rejected channel has a more equal distribution of the spectral power.

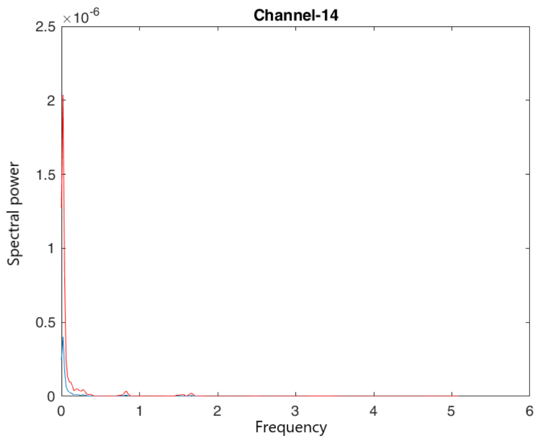


Figure 10: An example of an included channel.

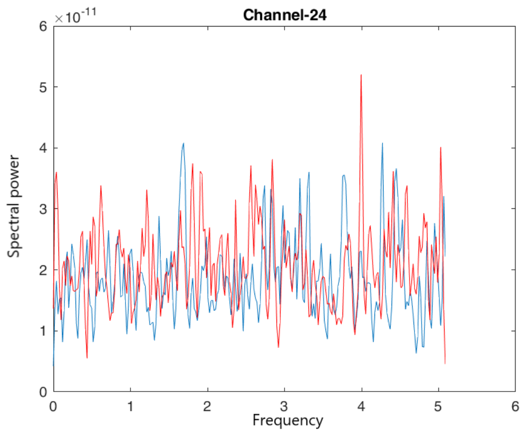


Figure 11: An example of a rejected channel.

The automatic channel rejection step within the pipeline bases on an algorithm called “pwelch” which calculates the PSD. This function bases on Welch’s power spectral density estimate which calculates the power of the signal at different frequencies (Welch, 1967). The function for channel rejection is borrowed from the open-access source code developed by the fNIRS research group from the University of California (Burns, 2018).

The parameters used for channel rejection are the quartile coefficient of dispersion (QCoD) and a saturation window length. The QCoD is a statistical measure that compares the 25-percent quartile to the 75-percent quartile of the spectral data for each channel (Bonett, 2006) (Dieffenbach et al., 2020). The saturation length bases on the maximum time of saturation within a given time. The employed saturation length is equal to the default length of 2 seconds used within the function (Burns, 2018). These parameters are also verified in this work by visual analysis of the spectral plots for each channel within the test data.

Step 6: Short channel removal

Step 6 includes the removal of the data in the short channels that are labeled in step 3. This step is employed as the short separation data is not included in this test. However, this step should be excluded if the experimental design requires data from the short separation channels.

Step 7: Intensity to optical density

One of the necessary functions in fNIRS processing is the conversion from raw intensity data within each channel, to the optical density (Pfeifer et al., 2017). The optical density, also known as the optical absorbance of a material, is the logarithmic intensity ratio between the light transmitted through the tissue and the light received after the absorption. Equation 1 describes the relationship between these two intensities (Zhang & Hoshino, 2019).

$$A(\lambda) = -\log_{10}\left(\frac{I_0}{I_1}\right) \tag{1}$$

For fNIRS measurements, the optical absorption occurs within the different layers of tissue as the light travels through the outer region of the brain. As illustrated in figure 3, the transmitted light has a banana-shaped path and is affected by different factors that need to be considered.

Step 8: Optical density to concentrations

For deriving the information about the deoxy- and oxygenated hemoglobin from the optical density data, conversion to hemoglobin concentrations is necessary. This is done by using the modified Beer-Lambert law (mBLL) to convert the data obtained by the two wavelengths into relative changes in concentrations (Pfeifer et al., 2017). An important aspect to consider when performing this conversion is the light scattering that occurs along the photon’s path within the tissue as illustrated in Figure 12.

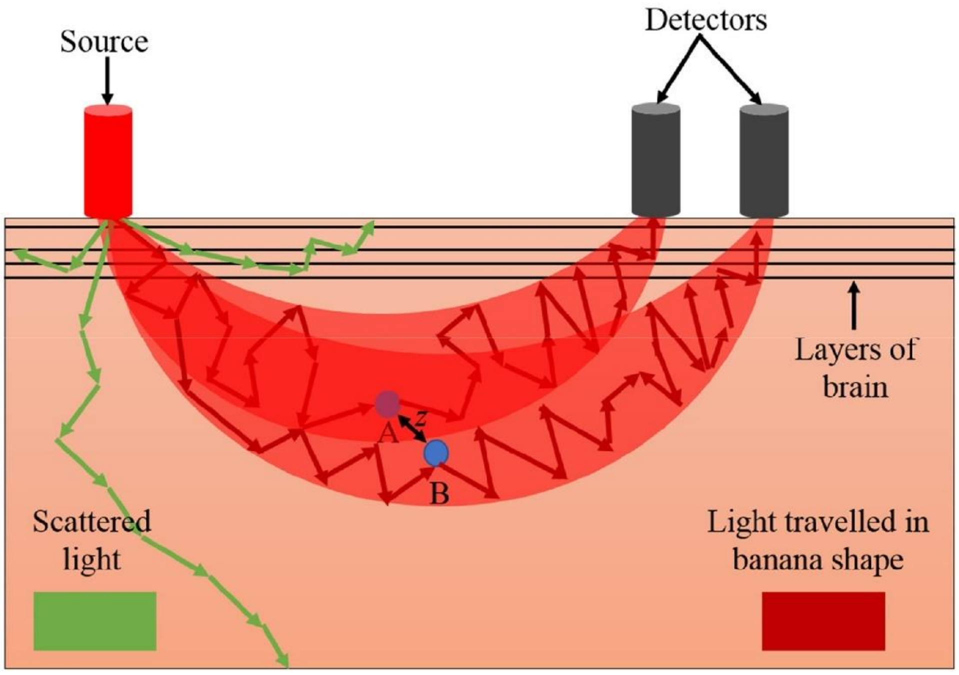


Figure 12: Illustration of the photon scattering through tissue. By Shoaib et al. (2019).

This scattering results in the photon having a longer path and being influenced by tissue that is outside of the region of interest. To compensate for this influence, a factor is implemented into the conversion. As several potential mathematical algorithms can be implemented, the factor is dependent on the available information regarding the participants of the study. The differential path length factor (DPF) is one option that takes the age, wavelength, and type of tissue of the participant into account (Scholkmann & Wolf, 2013). Another factor is the partial pathlength factor (PPF) which is the product of the DPF and the partial volume factor (PVF) (Whiteman et al., 2018). However, these factors are seldom precise in fNIRS processing (NIRx Medical Technologies, 2020)

The utilized function within the script is the "nirs.modules.BeerLambertLaw" which is based on the approach from Jacques (2013) which employs the PPF of 0.1 for the compensation of the scattering.

Step 9: Bandpass filtering

To remove unwanted noise within the data it is necessary to apply an algorithm that excludes the frequencies outside of the frequency area of interest. Without any golden standard, this is a frequently discussed topic within the fNIRS community (Yucel et al., 2021). However, Dans et al. (2021) report frequency filters to have been the most employed pre-processing technique in motor control studies in the last decade and are thus employed in the current pipeline.

To ensure that no important frequencies relevant for cortical activity are removed, a 4th order IIR Butterworth filter is applied with a lower cutoff frequency of 0.01 Hz and an upper cutoff frequency of 0.5 Hz (Dans et al., 2021). This bases on the existing bandpass filter function within the toolbox called "eeg.modules.bandpass" (Santosa et al., 2018).

Step 10: Reorganization of the channels

To facilitate the plotting of the functional connectivity within the experimental data, it is necessary to reorganize the channels by means of their position on the scalp. As the channels within the .nirs format are ordered by their channel number, which is independent of their position on the head, it is necessary to reposition them within the data matrix. The new order of the channel data is based on an alphabetic order of their respective Brodmann area (Zilles, 2018).

Step 11: Deleting short separation channels

To exclude the empty channels, that previously consisted of the data from the short separation channels, their column within the data is removed from the data matrix. The new data is thus only containing the information from the long separation channels.

3.2.3 Processing

The two processing steps taken in the pipeline are two prominent techniques used in the processing of fNIRS data. However, the GLM is only included to enable further implementation.

Step 12: GLM

As the main aim of this thesis is to explore functional connectivity, general linear model analysis is outside of the scope of this work. However, to facilitate further implementation of the GLM analysis within the pipeline, a GLM processing algorithm is included in the scripts to illustrate how it processes the data. This is chosen based on the broad use of GLM analysis in fNIRS motor control studies and as it can work as a framework for future development of the pipeline (Dans et al., 2021).

Another reason for implementing this in this current pipeline version is that it can present interesting information regarding the activation of the different areas. The activation is plotted based on the t-statistics with a p-value < 0.05. These parameters are chosen to plot the statistically significant activations of the different areas based on the stimuli information. The statistically significant channel activations that are found after the GLM analysis can then be employed in seed-based functional connectivity analysis. However, the current functional connectivity analysis is based on a whole-brain approach and the utilization of the GLM within the pipeline should be further investigated.

Step 13: Functional connectivity

The objective of this thesis has been to develop a pipeline for functional connectivity analysis of motor control studies by using a whole-brain approach. As fNIRS functional connectivity analysis both can be done in the time domain and the frequency domain, it is possible to apply different algorithms in this approach. In this work, a time-domain correlation is utilized to enable functional connectivity analysis for temporal correlations between brain areas.

Without any clear state-of-the-art methods for such analysis, the pipeline is developed employing an algorithm that is known for its robustness and use of the Pearson correlation. The function employed is called "nirs.sFC.ar_corr" which bases on a pre-whitening autoregressive correlation model (Santosa et al., 2018). To ensure a robust analysis, the input of the function sets the robust flag to "true" and leads the function to employ the "nirs.math.robust_corrcoef2" for robust correlation analysis. This function bases on a robust correlation coefficients approach by Shevlyakov and Smirnov (2011) that is frequently employed in the Brain AnalyzIR toolbox. The pre-whitening properties of the function reduce the autocorrelation in the data by removing the serially correlated data based on the oversampling of the hemodynamic response data (NIRx Medical Technologies, 2020). The technique employed in this algorithm is further explained in Barker et al. (2013).

The outputted correlation values calculated by the FC function come in the form of an adjacency matrix that illustrates the connectivity between the channels. The adjacency matrix is an N-by-N matrix where N is the channel number based on the alphabetic order of the brain regions.

After the functional connectivity algorithm is applied, the correlated data is plotted in a circular graph. This graph bases on the "circularGraph" open-source MATLAB function created by Kassebaum (2022). Here, the adjacency matrix is converted into a network model where the nodes in the circular plot represent the channels within the data and the edges between them represent their respective correlation value. The magnitude of the correlation value is illustrated by the width of the edges in the plot.

To enable analysis based on an increase in the correlation values, the adjacency matrix is thresholded over several iterations. The iterations start at a 50% correlation and are stepwise increased by 5% for each cycle. This value is chosen to enable an analysis of the strong positive correlations between the channels (Nettleton, 2014). Both the adjacency matrix and the circular graph are plotted for each iteration.

Step 14: Save workspace and generate the report

In the last step of the pipeline, the workspace variables are stored alongside the report's completion. The variables are stored as a .mat file containing all the variables and structures generated along the way to enable further analysis of the data or the control check of the different steps taken during the processing. To facilitate analysis that adheres to the recommendations for fNIRS practice and studies published in Yucel et al. (2021), the report is generated alongside individual storage of all the figures. This ensures that the user can perform a visual inspection of the output from each step and be able to extract relevant figures for the paper.

Another aspect of good practice in fNIRS studies is to give detailed descriptions of the pre-processing and processing steps employed when generating the data (Yucel et al., 2021). Thus, the pipeline produces a written presentation of the variables, coefficients, and techniques used during the process. As these are also provided in the workspace file, the presented information is selected based on the importance related to the study. This information is presented on the last page of the report alongside the description of the abbreviations included in the circular graph and the excluded channels.

3.3 The prior Gaitline experiment

The data used to illustrate the workings of the pipeline is obtained in an ongoing study by Oslo Metropolitan University and Gaitline AS. The experiment is part of the doctoral thesis work of Ph.D. Candidate Haroon Khan and the data are used with his and his supervisors' permission. The experiment is conducted separately from the work on my thesis, and it is important to clarify that I have not had any role in the experimental design or the data collection process. The following sections include a presentation of the instrumentation used during data collection and a brief description of the experimental paradigm. This is presented to provide a basic understanding of the context of the experimental data. However, the rationale behind the study is not provided as it is outside of the scope of this thesis.

3.3.1 Experimental design

With an aim of "investigating the human gait patterns with different footwear conditions while doing infinity walk", the study used fNIRS to assess the hemodynamics of the participants during the experiment (Khan, 2022). The experimental paradigm consists of initial rest, task, rest, and final rest. Figure 13 illustrates the process. The time points for the onset and offset of the stimuli are recorded and are found in the data.

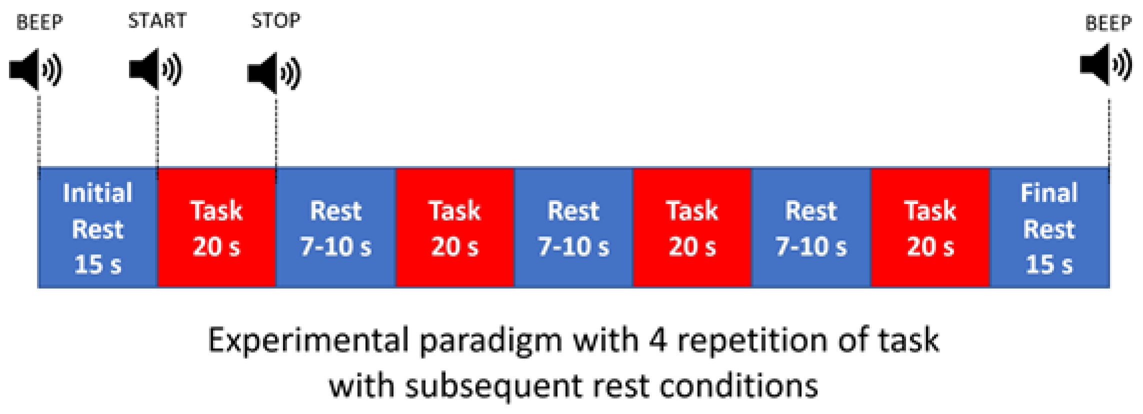


Figure 13: The experimental paradigm. By Khan (2022).

The paradigm is conducted for each of the 5 conditions presented in Table 2.

Table 2: The conditions of the Gaitline study. The information is taken from Khan (2022).

Condition nr.	**Description**
1	Barefooted walk on the plane surface of the Gaitline research lab.
2	Walk with flat sole sandals.
3	Walk with the calculated wedge angle of pronation and put the corrected wedge in a flat sandal wedge.
4	Walk with participant's personal shoes.
5	Walk with Gaitline shoes

Infinity walk, illustrated in Figure 14, is a walking pattern that requires the participant to perform complex gait behavior and is a recognized technique used in gait and balance studies (Szymanski, 1997).

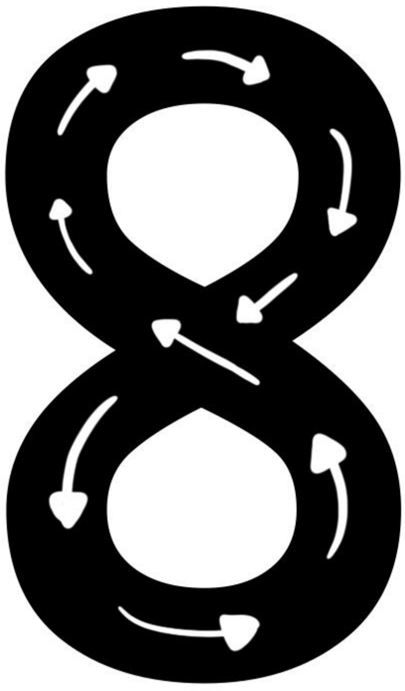


Figure 14: Illustration of infinity walk.

3.3.2 Experimental details

The instrumentation used for data acquisition is the NIRSport 2 mobile fNIRS system delivered by NIRx Medical Technologies, Germany. The device utilized wavelengths of 760 nm and 850 nm during acquisition and the current optode montage is based on a standard motor cortex montage. The study consisted of 6 participants who all suffered from different degrees of pronation (Khan, 2022). A further description of the research is presented in the application to REK presented in Appendix C.

3.4 Ethical considerations

The work in this thesis is connected to the research project at Oslo Metropolitan University led by Professor Peyman Mirtaheri. As presented in Appendix C, the project is approved by REK under application number 322236. As the main goal of the project is to investigate the potential of fNIRS technology, there are no additional requirements. Data gathering was consent-based, and all participants consented.

As the pipeline is handling sensitive biomedical health information, the privacy of the participants is important to handle. Measures regarding privacy are thus important to consider. However, the data used in this thesis is presented anonymously and only for illustrational purposes.

4 Results

The main objective of this thesis has been to develop a pipeline for fNIRS functional connectivity analysis of motor control studies. Thus, the main results of this thesis evolve around the workings of the pipeline and how it can be implemented in future studies. This section is therefore including the presentation of the output from the pipeline followed by an example of how to interpret the results.

4.1 The output of the pipeline

The pipeline generates a report that includes all the plots that are produced and relevant information from the pre-processing and processing steps. In parallel, each plot is stored separately in a folder with the same placement as the report. An example of the complete report is provided in Appendix E and shows the output from one of the tests performed in the Gaitline study. However, examples of some of the most important plots are presented in this section as well. All the following plots are from the same experimental trial. This is provided to give insights into what output the user receives, and how these plots can be interpreted.

Figures 15 and 16 below illustrate the raw fNIRS data in the time domain against the raw data where the 5 first seconds and the last 15 seconds are removed from the data. The graph below the raw data represents the start and stop of the stimuli.

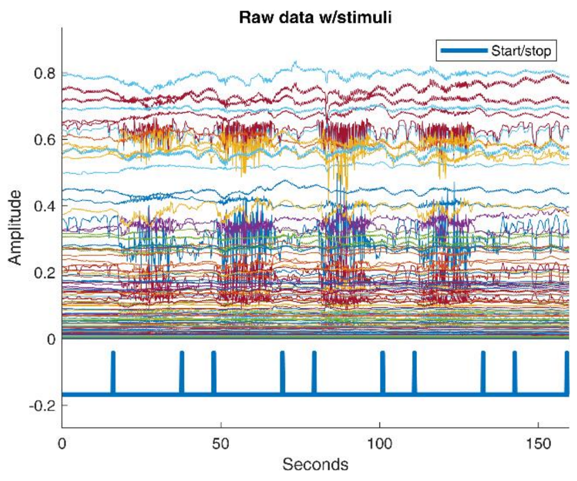


Figure 15: An example of how the raw experimental data is plotted.

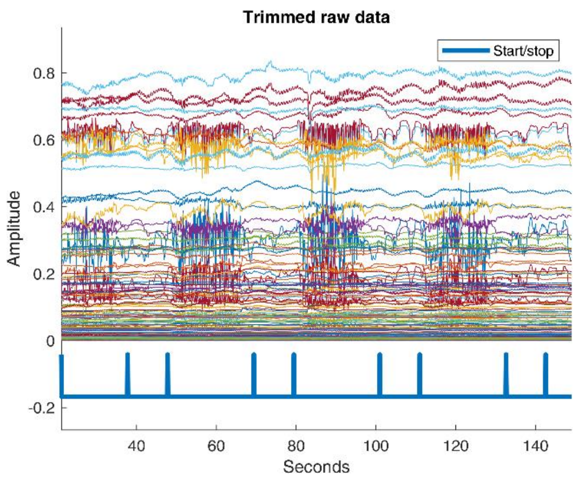


Figure 16: An example of how the trimmed experimental data is plotted.

Figure 15 and 16 clearly illustrates how the data is distorted during the stimuli. As the stimuli are based on movement, the fluctuation in the channels is most likely connected to motion artifacts. As the fNIRS measurements are prone to such motion artifacts, it is necessary to compensate for the baseline shift that can occur from them. This compensation is conducted according to the recommendations from Dans et al. (2021) and is done for each channel that qualifies for such adjustments. Figure 17 below illustrates this baseline shift, alongside the plotting of the “good” channels presented in Figure 18.

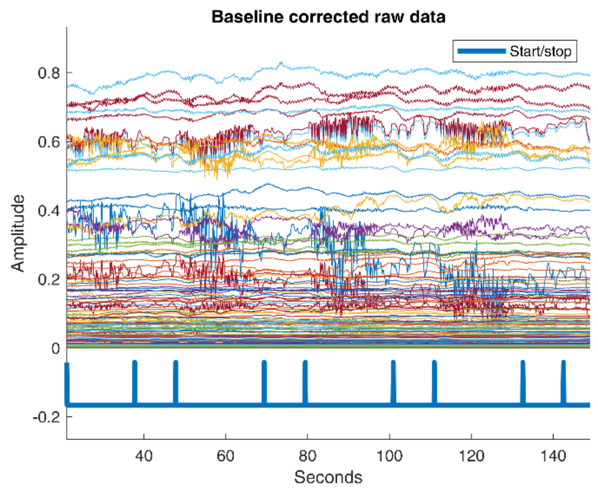


Figure 17: An example of baseline-corrected data.

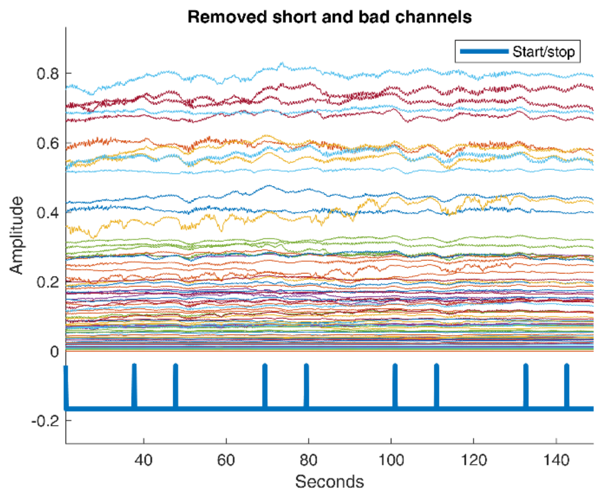


Figure 18: An example of data without 'bad' and short channels.

After the channel removal is conducted in steps 5 and 6, the conversion from intensity to optical density is performed. Figure 19 presents what the converted plots look like. The data is now centered around zero, where each channel fluctuates based on the change in optical density.

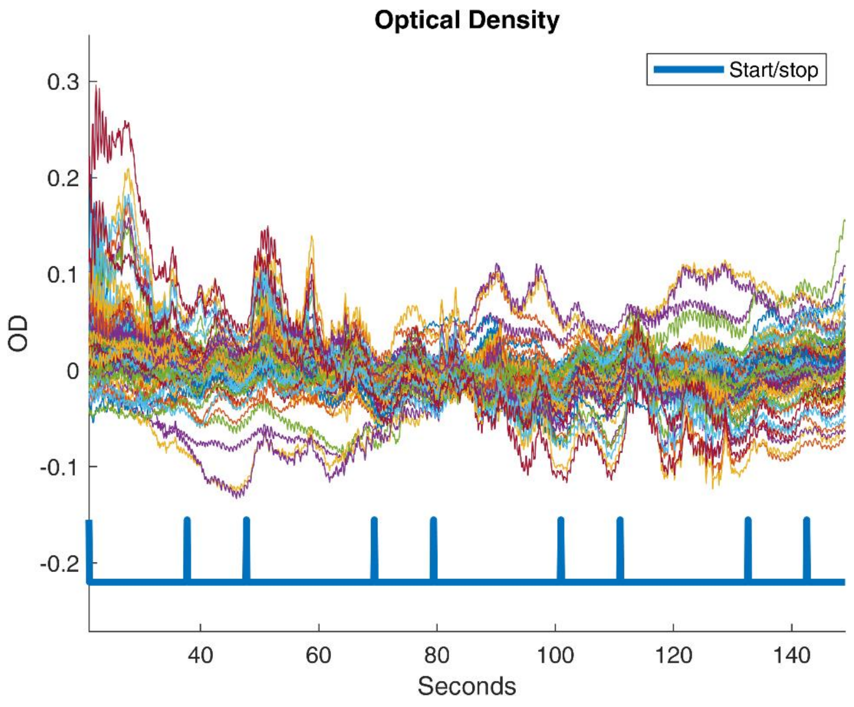


Figure 19: An example of the optical density plot.

After the optical density data is derived, the conversion to hemoglobin concentration is performed. These concentrations are presented in Figure 20 which plots the graphs for both deoxy- and oxygenated channels. Figure 21 presents the filtered concentrations data. The differences between the plots are seen in the smoothing of the signals as the frequencies beyond 0.5 Hz are removed. The lower cutoff frequency of 0.01 Hz removes some of the longer fluctuations in the signals.

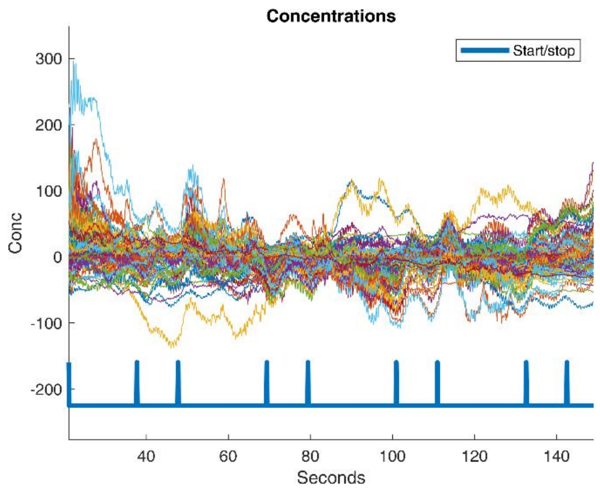


Figure 20: An example of the plotted concentration data.

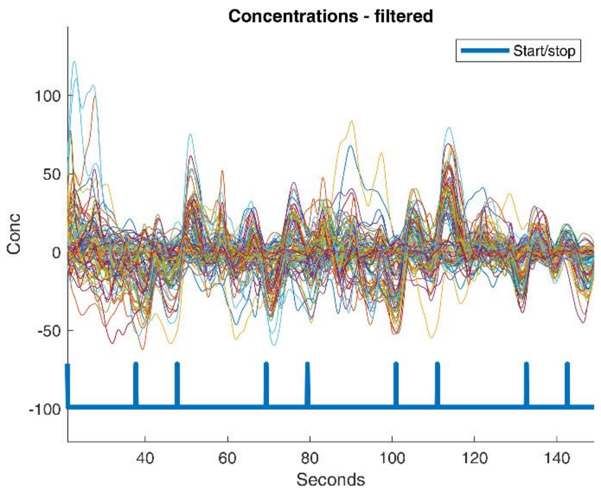


Figure 21: An example of the filtered concentration data.

The plots of the hemoglobin concentrations are the last plots produced during the pre-processing steps of the pipeline. After this phase, the processing steps are conducted parallelly as presented in the overview of the pipeline in Figure 9. These steps include general linear modeling and functional connectivity processing, and their respective plots are produced. However, as the GLM is included in the pipeline to mainly facilitate future studies, the functional connectivity analysis is the focus of the current analysis.

Figure 22 presents the adjacency matrix, also known as the connectivity matrix, which is the result of the functional connectivity analysis.

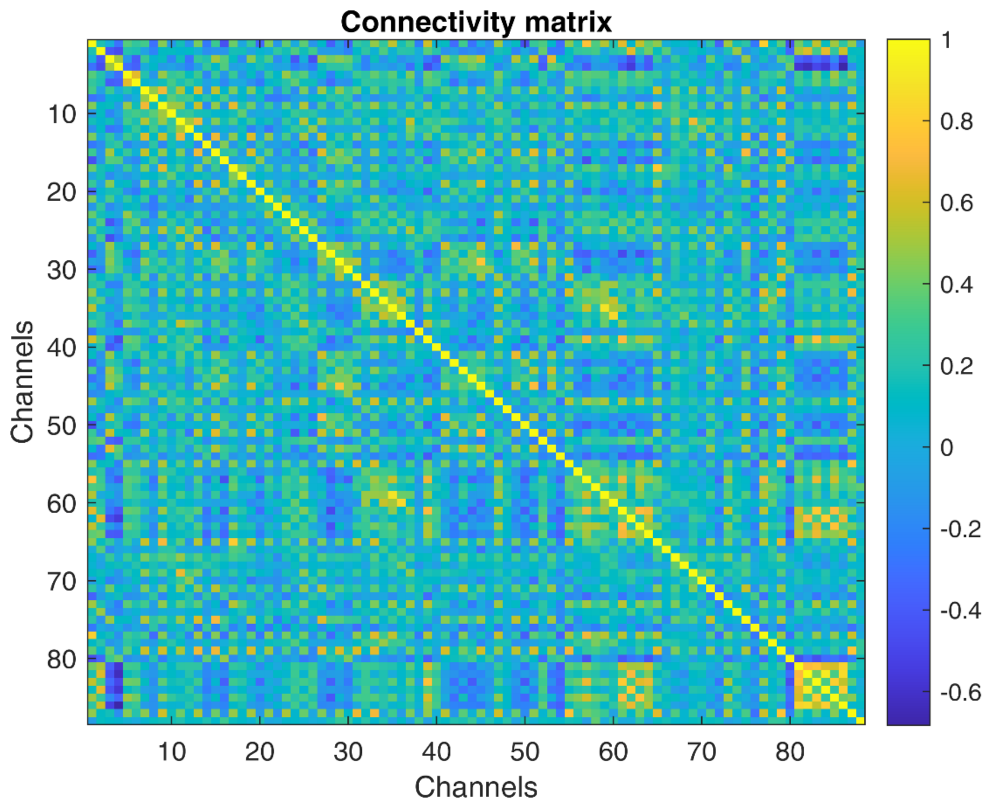


Figure 22: An example of the adjacency/connectivity matrix from the functional connectivity analysis.

The plot displays the level of connectivity, in form of the Pearson's R-value where 1 represents 100% positive correlation between the channels, and -1 represents 100% negative correlation. Zero indicates that there is no correlation at all. The diagonal yellow line illustrates the correlation the channel has with itself and will always have a 100% correlation for this pipeline. The numbers of the channels are based on the alphabetic order of the brain regions.

As one of the objectives of this thesis is to present the functional connectivity data using a circular graph, the number of connections is advantageously reduced to improve the interpretability of the circular graph. This is done by stepwise thresholding the adjacency matrix and plotting it for each 5% increment in connectivity starting at 50% correlation. Figure 23 illustrates an adjacency matrix with a 75% correlation between the channels. This value is chosen based on that a 75% percent correlation is usually known as a very strong correlation in time series data (Kozak, 2009).

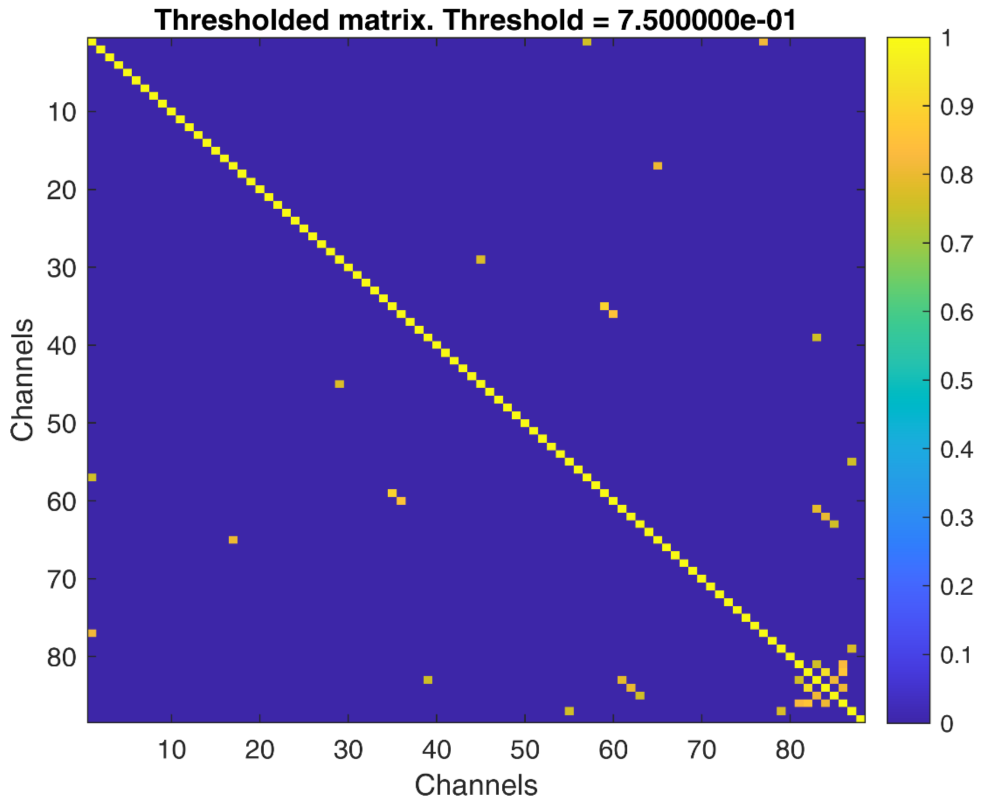


Figure 23: An example of a thresholded adjacency matrix with a threshold of 75% positive correlation.

As shown in Figure 23, the correlated channels are heavily reduced. This enables a more interpretable connectivity plot in the circular graph as the number of connections is reduced.

To explore the connectivity between the different regions of the brain, the circular graph in Figure 24 shows the regions that have strongly correlating behavior. The abbreviations used in the plot are further explained in the last pages of the example report provided in Appendix E.

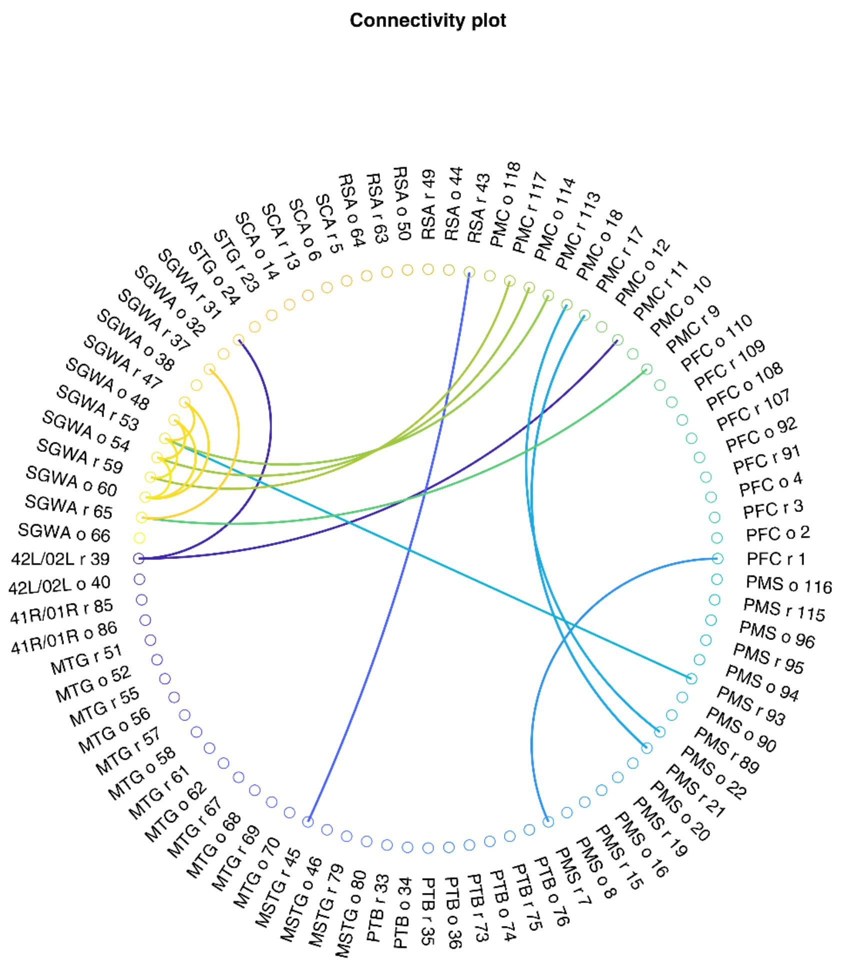


Figure 24: An example of the connectivity plot (circular graph). The threshold is here 75% positive correlation.

As illustrated in Figure 24, the connectivity between the brain regions (nodes) of the network is represented by the graphs (edges) between them. With the relatively high threshold of 75% correlation, only the strongest correlations are plotted. The colors of the edges are not connected to the correlation values as it only represents the area of connection. The correlation value is represented by the linewidth of the edges but is not easily distinguishable in high correlation plots.

4.2 FC analysis of experimental data – an example

In this section, an analysis of experimental data from one participant is conducted to illustrate how the pipeline's output can facilitate research. The analysis bases on the functional connectivity plots from the cortical activity data from one participant during different conditions. This analysis is provided for illustrational purposes and the experimental data will further be analyzed in the Doctoral Thesis of Ph.D. Candidate H. Khan.

The following analysis is based on a region of interest approach which compares all regions against each other. The plotting is conducted based on the HbO_2/Hb time series that has been filtered with a fourth-order Butterworth bandpass filter with the cutoff frequencies of 0.01-0.5 Hz. Additional information about the parameters and methods is found in the report in Appendix E.

The following figures show the differences in connectivity from the different conditions of the experiments. All the plots are from the same participant and employ a 75% correlation threshold. This is chosen based on the relatively strong correlation represented by the 75% correlation (R > 0.75) threshold. The participant is randomly chosen.

4.2.1 Flat vs wedged sandal

Figures 25 and 26 illustrate the differences between condition 2 (walking with flat sole sandals) and condition 3 (walking on wedged heel sandals). These conditions are chosen to assess the connectivity differences based on different wedge angles of the calcaneus.

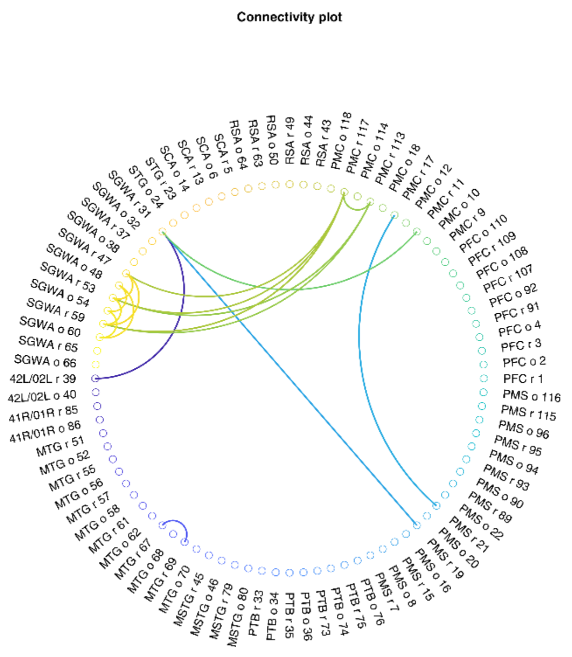


Figure 25: The connectivity plot of participant 3 from condition 2 (flat sole) data with R-threshold = 75%.

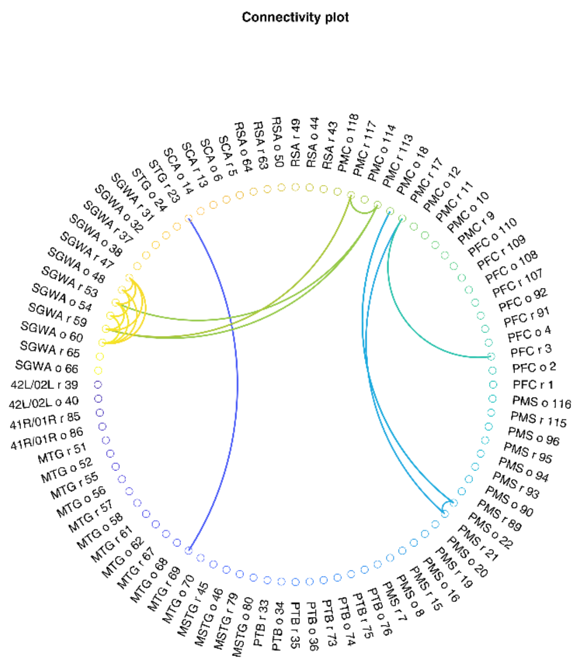


Figure 26: The connectivity plot of participant 3 from condition 3 (wedged hell) data with R-threshold = 75%.

Both similarities and differences can be observed between condition 2 and condition 3. Similarities in the connectivity can be seen as some of the same channels show correlations over 75%. For example, correlations between the channels within the Supramarginal Gyrus of Wernicke’s area (SGWA) show high correlations in both the oxygenated and the deoxygenated channels. These deoxygenated channels are also showing a high correlation to the deoxygenated channels in the Primary Motor Cortex (PMC) during both conditions.

Connectivity between channel PMS r 21 (deoxygenated channel in pre-motor and supplementary motor cortex) and channel PMC r 17 (deoxygenated channel in primary motor cortex) is shown to be strong in both plots. However, their respective oxygenated channels are only showing a correlation of over 75% during condition 3.

Another difference is observed as the connection between the PMS r 19 (deoxygenated channel in Pre-motor and Supplementary Motor Cortex) and SGWA r 31 (deoxygenated channels in Supramarginal Gyrus of Wernicke’s Area) only occurs during condition 2. Inversely, the correlation between the STG and MTG is only observed when walking with a wedged heel. Another remark is that the majority of the correlating channels are the deoxygenated channels.

4.2.2 Barefoot vs wedged Gaitline shoe

Figures 27 and 28 present the connectivity during condition 1 (walking barefoot) and condition 5 (walking with Gaitline shoes). By comparing the connectivity between these conditions, information about the potential impact of the Gaitline shoes can be assessed.

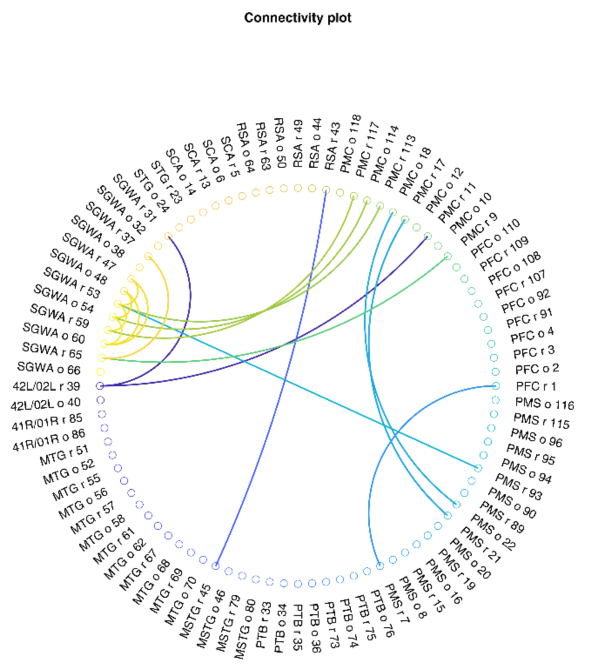


Figure 27: The connectivity plot of participant 3 during condition 1 (barefoot) data with R-threshold = 75%.

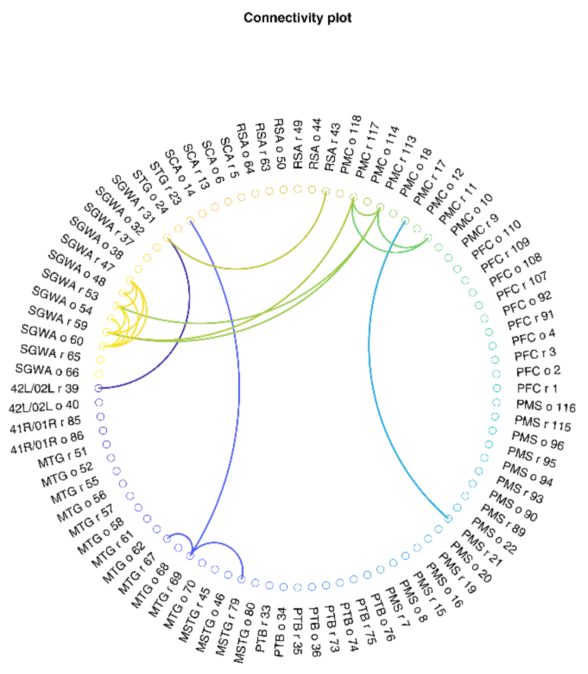


Figure 28: The connectivity plot of participant 3 during condition 5 (Gaitline shoe) data with threshold = 75%.

Similar to the previous conditions a strong correlation between the PMS and the PMC is observed together with the correlation between PMC and SGWA. The majority of correlation between the deoxygenated channels compared to the oxygenated ones is also present during these conditions.

The plot from the barefooted condition shows a stronger correlation between the PMS and the SGWA than the condition using Gaitline shoes.

A comparison of the functional connectivity can be conducted with several approaches. By comparing the FC between other conditions, assessment of different footwear and calcaneus wedging may be addressed. Moreover, assessments between participants during the same conditions can also provide interesting information. However, as these comparisons are provided to illustrate the workings of the pipeline, further analysis of the data from the experiment will be addressed in future studies.

5 Discussion

With the objective of developing a pipeline for functional connectivity analysis of fNIRS motor control studies, this thesis has pointed toward several different concepts and areas that need to be addressed. The final product from this work is a functioning pipeline with the ability to streamline the pre-processing and processing step which can significantly reduce the time related to this stage in research. To provide a solid foundation for the design of the pipeline, the following research questions were raised:

1. What is the state-of-the-art procedure for analyzing motor control fNIRS data?
2. What features are most important to include in a pipeline for functional connectivity analysis of fNIRS data from motor control studies?

Without any clear consensus within the field of fNIRS research, none of the research questions were given conclusive answers. The design of the pipeline was thereby based on the most promising techniques found in the literature. However, the following sections include evaluations regarding the different aspects of the pipeline and the main findings in this work. This is provided to give important insights behind this work and to help others avoid pitfalls.

5.1 The development process

In the initial stage of the development of the pipeline, the different possibilities of its design were explored. Three alternatives were found after reviewing the state-of-the-art literature and software: 1. Create a pipeline within a stand-alone software, 2. Develop a pipeline based on existing toolboxes, and 3. Develop a completely new repository of necessary functions.

The first alternative enables a fast development process as it has premade functions for fNIRS pre-processing and processing. These allow the user to implement the desired steps into the pipeline and can provide a given set of data. However, the flexibility of the usage is limited as only a few processing steps are available to implement. Also, the lack of access to the utilized scripts from the functions reduces the transparency of the pipeline. As it is important to present the processing techniques used on the data, this alternative was regarded as suboptimal (Yucel et al., 2021).

The third alternative included the development of a whole new repository to facilitate a pipeline. This would enable a fully transparent pipeline with functionalities designed for the current goal. However, the complexity of such a process was found to be disproportionate to the given time frame for this thesis.

With several existing MATLAB toolboxes for fNIRS data processing, the second alternative was selected for the development of the pipeline. As the majority of well-known toolboxes are MATLAB-based (presented in Table 1) MATLAB was chosen as the platform for further implementation. The different toolboxes provide reliable functions for different approaches to fNIRS processing and are known for their utilization in several studies. However, the implementation of these toolboxes can be challenging and time-consuming. After reviewing potential toolboxes to implement, the HomER3 toolbox by Huppert et al. (2009) was found to have the functionality required for the current objective. Alongside a broad variety of pre-processing and processing functions available, its capabilities of processing the shared NIRS file format (.snirf) contributed to it being selected.

Nevertheless, after several iterations of developing different versions of the pipeline, challenges related to its useability made it necessary to investigate other solutions. This led to the investigation of how to utilize the Brain AnalyzIR toolbox by Santosa et al. (2018) in the pipeline instead. Even though the toolbox is in constant development, it proved to have the relevant functions and compatibility necessary for creating a pipeline although some functions needed slight modifications to work. Even though the Brain AnalyzIR toolbox was used in this toolbox, a deeper comparison of its functionalities could have benefitted the development process.

After generating it as a live script format in MATLAB, the pipeline was implemented in a MATLAB-based GUI. The environment for creating the interface allows the development of an interactive user experience, but the application designer platform has several limitations regarding design possibilities. This led to a user interface with basic functionality and design. The available user interactions are either a simple button interface or the actual script. Both options enable the user to access the features of the pipeline, but there is no easy way of changing its parameters or workflow. The GUI does not enable the user to change any of the settings and changing the scripts require programming experience. By adding these options to the two interfaces, the pipeline could have had more flexibility. The GUI relies on correct use by the user and could benefit from additional “fail-safe”-functions to prevent program failure. However, the necessary functionality for achieving the objective of having a functioning pipeline was implemented and the GUI allows the user to run the pipeline.

5.2 The pipeline

The pipeline is designed to enable effective and robust functional connectivity analysis of fNIRS motor control studies. This is done by implementing the most promising techniques found in state-of-the-art literature within the field. As the review present that there is no golden standard to base the design on, it is not achievable to provide a conclusive solution to the processing steps. The pipeline is therefore based on techniques that show great potential when processing motor control fNIRS data.

5.2.1 Pre-processing

The inconsequent use of pre-processing steps within the field makes it challenging to choose what steps to implement. The chosen steps are therefore based on recommendations found in the literature, available functions within the toolbox, and experience exchanges within the research team at the university. To increase the validity of these decisions, a thorough review of the produced results from all available processing functions should be performed.

One of the steps that could benefit from such review studies is the channel rejection step. The channel rejection function bases on the automatic analysis of the power spectral density by employing a given saturation period and a QCoD threshold. The value of these parameters is based on what is used in similar studies (Burns, 2018). This could lead to too many or too few channels being rejected. By conducting further analysis of the impact of these two parameters, an even more robust solution could have been implemented. This could be done alongside addressing the impact of other channel rejection techniques.

The removal of motion artifacts is done by utilizing the commonly used step of baseline correction (Dans et al., 2021). Its effects can be seen by comparing the plots before and after baseline correction and correcting the baseline of the signals containing significant baseline shifts. However, as a common step in fNIRS data pre-processing is to manually remove temporal data that contains bigger motion artifacts, automatic removal of such artifacts could benefit the integrity of the results.

Other methods are also possible to implement or replace already utilized functions and would also beneficially be reviewed against each other. However, common guidelines for such decisions are an important contribution that the field of fNIRS is missing.

5.2.2 Processing for functional connectivity

Although the pre-processing steps employed in the pipeline are essential for the integrity of the data, the processing steps need to be equally reliable. Integrity is aimed to be prevailed in the processing step by implementing a processing algorithm for functional connectivity analysis with features that are known to be reliable within the literature.

The algorithm employed in the pipeline bases on a pre-whitening approach that removes temporal autocorrelation before calculating the Pearson's R correlation. This technique has shown to be important in resting-state FC fNIRS studies as it can remove autocorrelations (Blanco et al., 2018). Autocorrelations within the data can lead to high levels of false positives in the correlation and are thus important to remove to prevail the integrity of the data (Pinti et al., 2018). With other existing methods for calculating the correlation, a thorough comparison of their impact on the results would further strengthen the reliability of the pipeline.

With only one method for connectivity implemented, false positives within the data might be wrongly perceived as true positives as there is no analysis to compare the results to. This could lead to a wrong interpretation of the data. The pre-whitening of the signals aims to remove most of these autocorrelations, but the implementation of other connectivity methods could provide an even stronger foundation for correct analysis.

The functional connectivity analyses are mostly employed in resting-state studies, where the prevalence of motion artifacts is less than in motor control studies (Hu et al., 2020). Though the features within the pipeline aim to remove these artifacts, more studies regarding functional connectivity in task-evoked experimental paradigms would benefit the field.

The functional connectivity analysis is conducted on the signals within the 0.01 – 0.5 Hz frequency area. This area includes the areas related to several physiological processes. By separating the signals into smaller frequency bands before analysis, information about the connectivity within these bands might be derived. As presented in Yucel et al. (2021), such analysis could provide additional information about the connectivity based on their physiological characteristics.

Another relevant analytical approach to include could be to analyze the changes in connectivity during the different segments of the experimental paradigm. By separating the data into segments of “left turn”, “right turn”, “straight forward”, and “pause”, analysis between the segments could give additional information about the participant’s connectivity. This might provide further insights into how the brain responds to different footwear and gait patterns. This could technically be implemented by utilizing the already existing accelerometer and gyroscope data to differentiate the segments.

5.2.3 The output from the pipeline

The results generated by the pipeline aim to make the research process more effective by automating the processing pipeline and generating results for further analysis. This is done by facilitating functional connectivity analysis. By having the data created as a report based on the chronological plots from its process, the user has control of the data during the whole process.

The functional connectivity plots come in the form of an adjacency matrix without any threshold, an adjacency matrix with the thresholds, and a circular graph for all the thresholded adjacency matrices. As connectivity analyses often are based on only the adjacency matrix (NIRx Medical Technologies, 2019) the circular graph is providing an alternative way of presenting the data. By plotting the connections in the form of edges between the channels the data can be interpreted. The abbreviations for the brain areas are described within every generated report to support its interpretability. To further increase the interpretability of the connectivity analysis, plots of the optodes’ placement on the scalp could be added together with the connectivity edges between them.

Other graphical actions could also be taken to further develop the plots. However, such actions are inessential in this work as the circular graph includes the necessary information to perform the analysis.

5.3 The analysis of the Gaitline data

To present how the pipeline can be utilized for FC analysis, data from one participant in the Gaitline study were used as an example. Data from four conditions were assessed to address the impact of wedging of the calcaneus and the impact of the Gaitline shoes.

The comparison of a flat sole against a wedged sole indicated that similar connectivity occurred during both conditions. Both plots showed a strong correlation between the deoxygenated channels between the PMS and the PMC, alongside a strong correlation between the PMC and the SGWA. The same tendencies were observed during the barefooted walking and when using the Gaitline shoe. As previous studies have shown that the PMS and PMC are among the most active regions during walking (Herold et al., 2017), the findings regarding their temporal correlation are interesting to consider.

The stronger correlation between the PFC and the PMS during barefooted walk compared to walking with the Gaitline shoes is another indication that might be interesting to address further. As these areas are known for their activation during balancing tasks, their stronger correlating behavior might illustrate how the Gaitline shoes are impacting the users' connectivity when wearing them.

Regarding the skewed distribution of deoxygenated channels showing correlation levels above 75%, further assessment of their impact on the connectivity could provide a stronger foundation for the FC analysis. Montero-Hernandez et al. (2018) have already shown that this skew is expected when analyzing the functional connectivity using both hemoglobin species. Without any consensus in the field, they recommend including both species in the FC analysis to provide loss of information (Montero-Hernandez et al., 2018).

However, all the indications stated in this section are only based on one participant's data and can only work as an indication of what to address in future analysis.

6 Conclusion and future directions

With an aim of facilitating studies that try to map the functionality of the brain, the objective of this thesis has been to develop a processing pipeline for fNIRS motor control studies. With an ultimate goal of unraveling how the orchestration of the brain works, this program aims to contribute to the further development of the field by enabling researchers and students to perform reliable connectivity analysis. By reviewing state-of-the-art literature related to fNIRS data processing and functional connectivity analysis, this work aimed to discover the state-of-the-art steps needed to be implemented in such a pipeline. With no such standards discovered within the field, the pipeline is developed based on the most promising techniques employed in similar studies.

By mapping how the most utilized programs for fNIRS analysis are designed, this thesis found that analysis using MATLAB is common practice within the field. This led to the development of a MATLAB-based program utilizing the recognized NIRS Brain AnalyzIR toolbox by Santosa et al. (2018). The program enables automatic processing of experimental fNIRS data and generates a report consisting of the plots for each pre-processing and processing step taken. By providing both a user-friendly GUI version of the pipeline and a live script version, the user has the option of both conducting stepwise data processing or a fully automatic process.

One of the most promising techniques used in functional connectivity analyses was found to be autoregressive models that use pre-whitening of the signal to remove unwanted artifacts and to ensure reliable results. As the most used measure in functional connectivity analysis was found to be the Pearson's R coefficient of correlation, the model is implemented to perform a robust calculation of the correlation. The correlation between the channels is plotted both as an adjacency matrix and a circular graph to enable intuitive and interpretable analysis.

The workings of the pipeline are illustrated using data from a previous motor control experiment conducted by Oslo Metropolitan University and Gaitline AS. Analysis of some of the data illuminated how the connectivity plots provided by the pipeline can be used for functional connectivity analysis of motor control studies. These indications would be interesting to further analysis as they might lead to better rehabilitation for people in need. To adhere to the recommendations for best practices in fNIRS studies by Yucel et al. (2021), relevant information regarding the processing steps is implemented in the generated report.

With the lack of standardized pre-processing and processing steps in the field of fNIRS, conclusive answers to how a pipeline should be designed cannot be provided. More research regarding several important aspects is thus relevant to pursue.

During the work on this thesis, several gaps within the field were discovered. One of these gaps includes the lack of structural comparisons between existing toolboxes for fNIRS processing and analysis. By providing a thorough comparison between their functionalities and transparency, researchers and students would have a stronger foundation for conducting fNIRS research.

The low number of studies related to functional connectivity during motor control studies indicates that there is potential for further investigation in the field. Hopefully, the developed pipeline will both work as a helpful tool in such processing and also work as a framework that could inspire future studies. By constantly following the developments within the field, further development of this pipeline could be conducted. Natural future actions would be to compare the results generated in this pipeline to results from other FC tools, such as FC-NIRS by Xu et al. (2015). This could provide additional reliability to the outcomes of the pipeline.

The steps included within this pipeline are based on recommendations found in the literature. However, a thorough review of the differences caused by the implementation of different pre-processing and processing techniques would have a positive impact on the further development of the pipeline. By assessing the impact of interesting techniques such as DASAR, COFRE, MCLM, and SCAU as mentioned in Pinto-Orellana (2022), improvements in the analysis might occur. Implementation of promising machine learning techniques for better processing and feature extraction of cortical activation is another natural direction to pursue in the future of fNIRS motor control studies.

With the current focus of the research project within Oslo Metropolitan University regarding the exploration of a hybrid solution for fNIRS and EEG (Appendix C), functions for EEG processing should be implemented within the pipeline. Such processing functions are available in the NIRS Brain AnalyzIR toolbox and could give additional information regarding the spatiotemporal functional activity in the brain. Additional information might also be acquired by facilitating for processing of the short separation channel data and the IMU data from the NIRS Wing system. Moreover, by separating the experimental data into both temporal and spectral segments, analysis of the cortical activity might reveal important information about the complex human brain.

Functional Near-Infrared Technology is a neuroimaging technique showing great potential in the field of neuroimaging and further developments in its processing techniques can help us understand the orchestration of the human brain.

References

Aarabi, A., & Huppert, T. (2016). Characterization of the relative contributions from systemic physiological noise to whole-brain resting-state functional near-infrared spectroscopy data using single-channel independent component analysis. *Neurophotonics*, *3*(2), 025004. https://doi.org/10.1117/1.NPh.3.2.025004

Anwar, A. R., Muthalib, M., Perrey, S., Galka, A., Granert, O., Wolff, S., Heute, U., Deuschl, G., Raethjen, J., & Muthuraman, M. (2016). Effective Connectivity of Cortical Sensorimotor Networks During Finger Movement Tasks: A Simultaneous fNIRS, fMRI, EEG Study. *Brain Topography*, *29*(5), 645-660. https://doi.org/10.1007/s10548-016-0507-1

Barker, J. W., Aarabi, A., & Huppert, T. J. (2013). Autoregressive model based algorithm for correcting motion and serially correlated errors in fNIRS. *Biomedical Optics Express*, *4*(8), 1366-1379. https://doi.org/10.1364/BOE.4.001366

Blanco, B., Molnar, M., & Caballero-Gaudes, C. (2018). Effect of prewhitening in resting-state functional near-infrared spectroscopy data. *Neurophotonics*, *5*(04), 1. https://doi.org/10.1117/1.nph.5.4.040401

Bonett, D. G. (2006). Confidence interval for a coefficient of quartile variation. *Computational Statistics & Data Analysis*, *50*(11), 2953-2957. https://doi.org/https://doi.org/10.1016/j.csda.2005.05.007

Bonilauri, A., Sangiuliano Intra, F., Baselli, G., & Baglio, F. (2021). Assessment of fNIRS Signal Processing Pipelines: Towards Clinical Applications. *Applied Sciences*, *12*(1), 316. https://doi.org/10.3390/app12010316

Bowyer, S. M. (2016). Coherence a measure of the brain networks: past and present. *Neuropsychiatric Electrophysiology*, *2*(1). https://doi.org/10.1186/s40810-015-0015-7

BrainyQuote.com. (n.d.). *Michio Kaku Quotes*. BrainyQuote.com. Retrieved 14th of May from https://www.brainyquote.com/quotes/michio_kaku_615181

Bullmore, E. T., Fornito, A., & Zalesky, A. (2016). *Fundamentals of brain network analysis*. Elsevier.

Burns, S. (2018). *UCLA SCN preprocessing of fNIRS data*. In UCLA SCN. https://github.com/smburns47/preprocessingfNIRS/tree/master/Matlab

Cinciute, S. (2019). Translating the hemodynamic response: why focused interdisciplinary integration should matter for the future of functional neuroimaging. *PeerJ*, *7*, e6621. https://doi.org/10.7717/peerj.6621

Crosson, B., Ford, A., McGregor, K. M., Meinzer, M., Cheshkov, S., Li, X., Walker-Batson, D., & Briggs, R. W. (2010). Functional imaging and related techniques: an introduction for rehabilitation researchers. *Journal of rehabilitation research and development*, *47*(2), vii-xxxiv. https://doi.org/10.1682/jrrd.2010.02.0017

Dans, P. W., Foglia, S. D., & Nelson, A. J. (2021). Data Processing in Functional Near-Infrared Spectroscopy (fNIRS) Motor Control Research. *Brain Sci*, *11*(5). https://doi.org/10.3390/brainsci11050606

Dehghani, H., Eames, M. E., Yalavarthy, P. K., Davis, S. C., Srinivasan, S., Carpenter, C. M., Pogue, B. W., & Paulsen, K. D. (2009). Near infrared optical tomography using NIRFAST: Algorithm for numerical model and image reconstruction. *Communications in Numerical Methods in Engineering*, *25*(6), 711-732. https://doi.org/10.1002/cnm.1162

Dieffenbach, M. C., Gillespie, G. S. R., Burns, S. M., McCulloh, I. A., Ames, D. L., Dagher, M. M., Falk, E. B., & Lieberman, M. D. (2020). Neural reference groups: a synchrony-based classification approach for predicting attitudes using fNIRS. *Social Cognitive and Affective Neuroscience*, *16*(1-2), 117-128. https://doi.org/10.1093/scan/nsaa115

Eickhoff, S. B., & Müller, V. I. (2015). Functional Connectivity. In A. W. Toga (Ed.), *Brain Mapping* (pp. 187-201). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-397025-1.00212-8

Elam, J. S., Glasser, M. F., Harms, M. P., Sotiropoulos, S. N., Andersson,

J. L. R., Burgess, G. C., Curtiss, S. W., Oostenveld, R., Larson-Prior, L. J., Schoffelen, J.-M., Hodge, M. R., Cler, E. A., Marcus, D. M., Barch, D. M., Yacoub, E., Smith, S. M., Ugurbil, K., & Van Essen, D. C. (2021). The Human Connectome Project: A retrospective. *NeuroImage*, *244*, 118543. https://doi.org/https://doi.org/10.1016/j.neuroimage.2021.118543

Fantini, S., Frederick, B., & Sassaroli, A. (2018). Perspective: Prospects of non-invasive sensing of the human brain with diffuse optical imaging. *APL Photonics*, *3*(11), 110901. https://doi.org/10.1063/1.5038571

fNIRS.org. (2022, 13th of March 2022). *Shared Near Infrared Spectroscopy Format Specification*. The Sciety for functional Near Infrared Spectroscopy. Retrieved 5th of May from https://github.com/fNIRS/snirf/blob/master/snirf_specification.md#snirf-data-format-summary

Friston, K. J. (2011). Functional and effective connectivity: a review. *Brain Connect*, *1*(1), 13-36. https://doi.org/10.1089/brain.2011.0008

Glasser, M. F., Coalson, T. S., Robinson, E. C., Hacker, C. D., Harwell, J., Yacoub, E., Ugurbil, K., Andersson, J., Beckmann, C. F., Jenkinson, M., Smith, S. M., & Van Essen, D. C. (2016). A multi-modal parcellation of human cerebral cortex. *Nature*, *536*(7615), 171-178. https://doi.org/10.1038/nature18933

Gramfort, A., Luessi, M., Larson, E., Engemann, D., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., & Hämäläinen, M. (2013). MEG and EEG data analysis with MNE-Python [Methods]. *Frontiers in Neuroscience*, *7*. https://doi.org/10.3389/fnins.2013.00267

Hamacher, D., Herold, F., Wiegel, P., Hamacher, D., & Schega, L. (2015). Brain activity during walking: A systematic review. *Neuroscience & Biobehavioral Reviews*, *57*, 310-327. https://doi.org/10.1016/j.neubiorev.2015.08.002

Herold, F., Wiegel, P., Scholkmann, F., Thiers, A., Hamacher, D., & Schega, L. (2017). Functional near-infrared spectroscopy in movement science: a systematic review on cortical activity in postural and walking tasks. *Neurophotonics*, *4*(4), 041403. https://doi.org/10.1117/1.nph.4.4.041403

Hocke, L., Oni, I., Duszynski, C., Corrigan, A., Frederick, B., & Dunn, J. (2018). Automated Processing of fNIRS Data—A Visual Guide to the Pitfalls and Consequences. *Algorithms*, *11*(5), 67. https://doi.org/10.3390/a11050067

Hu, Z., Liu, G., Dong, Q., & Niu, H. (2020). Applications of Resting-State fNIRS in the Developing Brain: A Review From the Connectome Perspective. *Front Neurosci*, *14*, 476. https://doi.org/10.3389/fnins.2020.00476

Huppert, T. J., Diamond, S. G., Franceschini, M. A., & Boas, D. A. (2009). HomER: a review of time-series analysis methods for near-infrared spectroscopy of the brain. *Applied Optics*, *48*(10), D280. https://doi.org/10.1364/ao.48.00d280

Huppert, T. J., Hoge, R. D., Diamond, S. G., Franceschini, M. A., & Boas, D. A. (2006). A temporal comparison of BOLD, ASL, and NIRS hemodynamic responses to motor stimuli in adult humans. *NeuroImage*, *29*(2), 368-382. https://doi.org/10.1016/j.neuroimage.2005.08.065

Irani, F., Platek, S. M., Bunce, S., Ruocco, A. C., & Chute, D. (2007). Functional near infrared spectroscopy (fNIRS): an emerging neuroimaging technology with important applications for the study of brain disorders. *Clin Neuropsychol*, *21*(1), 9-37. https://doi.org/10.1080/13854040600910018

Jacques, S. L. (2013). Optical properties of biological tissues: a review. *Physics in Medicine and Biology*, *58*(11), R37-R61. https://doi.org/10.1088/0031-9155/58/11/r37

Kassebaum, P. (2022). *circularGraph*. In GitHub. https://github.com/paul-kassebaum-mathworks/circularGraph

Khan, H. (2022). *Experimental details of the Gaitline study* [Experimental design]. Oslo Metropolitan University.

Klein, F., & Kranczioch, C. (2019). Signal Processing in fNIRS: A Case for the Removal of Systemic Activity for Single Trial Data. *Front Hum Neurosci*, *13*, 331. https://doi.org/10.3389/fnhum.2019.00331

Kozak, M. (2009). What is Strong Correlation? *Teaching Statistics*, *31*(3), 85-86. https://doi.org/https://doi.org/10.1111/j.1467-9639.2009.00387.x

Latash, M. L. (2012). 1- A philosophical introduction. In M. L. Latash (Ed.), *Fundamentals of Motor Control* (pp. 1-4). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-415956-3.00001-4

Lau, T. M., Gwin, J. T., & Ferris, D. P. (2014). Walking reduces sensorimotor network connectivity compared to standing. *Journal of NeuroEngineering and Rehabilitation*, *11*(1), 14. https://doi.org/10.1186/1743-0003-11-14

Montero-Hernandez, S., Orihuela-Espina, F., Sucar, L., Pinti, P., Hamilton, A., Burgess, P., & Tachtsidis, I. (2018). Estimating Functional Connectivity Symmetry between Oxy- and Deoxy-Haemoglobin: Implications for fNIRS Connectivity Analysis. *Algorithms*, *11*(5), 70. https://doi.org/10.3390/a11050070

Monti, M. (2011). Statistical Analysis of fMRI Time-Series: A Critical Review of the GLM Approach [Review]. *Frontiers in Human Neuroscience*, *5*. https://doi.org/10.3389/fnhum.2011.00028

Muccigrosso, D., & Eggebrecht, A. (2018, 2018/04/03). NeuroDOT: A New Neuroimaging Toolbox for DOT.*OSA Technical Digest* Biophotonics Congress: Biomedical Optics Congress 2018 (Microscopy/Translational/Brain/OTS), Hollywood, Florida.

Nettleton, D. (2014). Chapter 6- Selection of Variables and Factor Derivation. In D. Nettleton (Ed.), *Commercial Data Mining* (pp. 79-104). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-12-416602-8.00006-6

NIRx Medical Technologies. (2019, 4th of April 2019). *NIRx Webinar Studying functional connectivity with fNIRS* [Webinar]. YouTube. https://www.youtube.com/watch?v=0lIrUszlBdI&t=12s&ab_channel=NIRxMedicalTechnologies

NIRx Medical Technologies. (2020, 24th of April 2020). *Introduction to NIRS Toolbox: Installation and Getting Started* [Webinar]. YouTube.

https://www.youtube.com/watch?v=zlAC3aWW8kc&ab_channel=NIRxMedicalTechnologies

Orihuela-Espina, F., Leff, D. R., James, D. R. C., Darzi, A. W., & Yang, G.-Z. (2017). Imperial College near infrared spectroscopy neuroimaging analysis framework. *Neurophotonics*, *5*(01), 1. https://doi.org/10.1117/1.nph.5.1.011011

Pfeifer, M. D., Scholkmann, F., & Labruyere, R. (2017). Signal Processing in Functional Near-Infrared Spectroscopy (fNIRS): Methodological Differences Lead to Different Statistical Results. *Front Hum Neurosci*, *11*, 641. https://doi.org/10.3389/fnhum.2017.00641

Pinti, P., Scholkmann, F., Hamilton, A., Burgess, P., & Tachtsidis, I. (2018). Current Status and Issues Regarding Pre-processing of fNIRS Neuroimaging Data: An Investigation of Diverse Signal Filtering Methods Within a General Linear Model Framework. *Front Hum Neurosci*, *12*, 505. https://doi.org/10.3389/fnhum.2018.00505

Pinti, P., Tachtsidis, I., Hamilton, A., Hirsch, J., Aichelburg, C., Gilbert, S., & Burgess, P. W. (2020). The present and future use of functional near-infrared spectroscopy (fNIRS) for cognitive neuroscience. *Annals of the New York Academy of Sciences*, *1464*(1), 5-29. https://doi.org/10.1111/nyas.13948

Pinto-Orellana, M. A. (2022). *Time-spectral modeling of biomedical signals* [Doctoral thesis, Oslo Metropolitan University].

Santosa, H., Zhai, X., Fishburn, F., & Huppert, T. (2018). The NIRS Brain AnalyzIR Toolbox. *Algorithms*, *11*(5). https://doi.org/10.3390/a11050073

Scholkmann, F., & Wolf, M. (2013). General equation for the differential pathlength factor of the frontal human head depending on wavelength and age. *Journal of Biomedical Optics*, *18*(10), 1-6. https://doi.org/10.1117/1.JBO.18.10.105004

Shevlyakov, G., & Smirnov, P. (2011). Robust Estimation of the Correlation Coefficient: An Attempt of Survey. *Austrian Journal of Statistics*, *40*, 147-156.

Shoaib, Z., Ahmad Kamran, M., Mannan, M. M. N., & Jeong, M. Y.

(2019). Approach to optimize 3-dimensional brain functional activation image with high resolution: a study on functional near-infrared spectroscopy. *Biomedical Optics Express*, *10*(9), 4684. https://doi.org/10.1364/boe.10.004684

Shumway-Cook, A., & Woollacott, M. H. (2017). *Motor control : translating research into clinical practice*.

Sporns, O., Tononi, G., & Kötter, R. (2005). The Human Connectome: A Structural Description of the Human Brain. *PLoS Computational Biology*, *1*(4), e42. https://doi.org/10.1371/journal.pcbi.0010042

Strangman, G., Franceschini, M. A., & Boas, D. A. (2003). Factors affecting the accuracy of near-infrared spectroscopy concentration calculations for focal changes in oxygenation parameters. *NeuroImage*, *18*(4), 865-879. https://doi.org/10.1016/s1053-8119(03)00021-1

Sutoko, S., Sato, H., Maki, A., Kiguchi, M., Hirabayashi, Y., Atsumori, H., Obata, A., Funane, T., & Katura, T. (2016). Tutorial on platform for optical topography analysis tools. *Neurophotonics*, *3*(1), 010801. https://doi.org/10.1117/1.NPh.3.1.010801

Szymanski, T. J. (1997). Infinity Walk: Preparing Your Mind to Learn! [Infinity Walk: Preparing your mind to learn!, D. Sunbeck]. *Research and Teaching in Developmental Education*, *13*(2), 113-115. http://www.jstor.org/stable/42801971

Tadel, F., Bock, E., Niso, G., Mosher, J. C., Cousineau, M., Pantazis, D., Leahy, R. M., & Baillet, S. (2019). MEG/EEG Group Analysis With Brainstorm [Methods]. *Frontiers in Neuroscience*, *13*. https://doi.org/10.3389/fnins.2019.00076

Tremblay, J., Martínez-Montes, E., Hüsser, A., Caron-Desrochers, L., Lepage, C., Pouliot, P., Vannasing, P., & Gallagher, A. (2022). LIONirs: flexible Matlab toolbox for fNIRS data analysis. *Journal of Neuroscience Methods*, *370*, 109487. https://doi.org/https://doi.org/10.1016/j.jneumeth.2022.109487

Van Essen, D. C. (2013). Cartography and Connectomes. *Neuron*, *80*(3), 775-790. https://doi.org/10.1016/j.neuron.2013.10.027

von Luhmann, A., Ortega-Martinez, A., Boas, D. A., & Yucel, M. A.

(2020). Using the General Linear Model to Improve Performance in fNIRS Single Trial Analysis and Classification: A Perspective. *Front Hum Neurosci*, *14*, 30. https://doi.org/10.3389/fnhum.2020.00030

Wang, J., Wang, X., Xia, M., Liao, X., Evans, A., & He, Y. (2015). GRETNA: a graph theoretical network analysis toolbox for imaging connectomics. *Front Hum Neurosci*, *9*, 386. https://doi.org/10.3389/fnhum.2015.00386

Welch, P. (1967). The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, *15*(2), 70-73. https://doi.org/10.1109/TAU.1967.1161901

Whiteman, A. C., Santosa, H., Chen, D. F., Perlman, S., & Huppert, T. (2018). Investigation of the sensitivity of functional near-infrared spectroscopy brain imaging to anatomical variations in 5- to 11-year-old children. *Neurophotonics*, *5*(1), 011009-011009. https://doi.org/10.1117/1.NPh.5.1.011009

Wong, J. K., Middlebrooks, E. H., Grewal, S. S., Almeida, L., Hess, C. W., & Okun, M. S. (2020). A Comprehensive Review of Brain Connectomics and Imaging to Improve Deep Brain Stimulation Outcomes. *Movement Disorders*, *35*(5), 741-751. https://doi.org/10.1002/mds.28045

Xu, J., Liu, X., Zhang, J., Li, Z., Wang, X., Fang, F., & Niu, H. (2015). FC-NIRS: A Functional Connectivity Analysis Tool for Near-Infrared Spectroscopy Data. *Biomed Res Int*, *2015*, 248724. https://doi.org/10.1155/2015/248724

Xu, Y., Graber, H. L., & Barbour, R. L. (2014, 2014/04/26). nirsLAB: A Computing Environment for fNIRS Neuroimaging Data Analysis.*OSA Technical Digest (online)* Biomedical Optics 2014, Miami, Florida.

Ye, J. C., Tak, S., Jang, K. E., Jung, J., & Jang, J. (2009). NIRS-SPM: Statistical parametric mapping for near-infrared spectroscopy. *NeuroImage*, *44*(2), 428-447. https://doi.org/https://doi.org/10.1016/j.neuroimage.2008.08.0

36
Yücel, M., Selb, J., Aasted, C., Petkov, M., Becerra, L., Borsook, D., & Boas, D. (2015). Short separation regression improves statistical significance and better localizes the hemodynamic response obtained by near-infrared spectroscopy for tasks with differing autonomic responses. *Neurophotonics*, *2*(3), 035005. https://doi.org/10.1117/1.NPh.2.3.035005
Yucel, M. A., Luhmann, A. V., Scholkmann, F., Gervain, J., Dan, I., Ayaz, H., Boas, D., Cooper, R. J., Culver, J., Elwell, C. E., Eggebrecht, A., Franceschini, M. A., Grova, C., Homae, F., Lesage, F., Obrig, H., Tachtsidis, I., Tak, S., Tong, Y., . . . Wolf, M. (2021). Best practices for fNIRS publications. *Neurophotonics*, *8*(1), 012101. https://doi.org/10.1117/1.NPh.8.1.012101
Yücel, M. A., Selb, J., Aasted, C. M., Lin, P.-Y., Borsook, D., Becerra, L., & Boas, D. A. (2016). Mayer waves reduce the accuracy of estimated hemodynamic response functions in functional near-infrared spectroscopy. *Biomedical Optics Express*, *7*(8), 3078. https://doi.org/10.1364/boe.7.003078
Yücel, M. A., Selb, J. J., Huppert, T. J., Franceschini, M. A., & Boas, D. A. (2017). Functional Near Infrared Spectroscopy: Enabling routine functional brain imaging. *Current Opinion in Biomedical Engineering*, *4*, 78-86. https://doi.org/10.1016/j.cobme.2017.09.011
Zhang, J. X. J., & Hoshino, K. (2019). Chapter 5 - Optical transducers: Optical molecular sensing and spectroscopy. In J. X. J. Zhang & K. Hoshino (Eds.), *Molecular Sensors and Nanodevices (Second Edition)* (pp. 231-309). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-814862-4.00005-3
Zilles, K. (2018). Brodmann: a pioneer of human brain mapping—his impact on concepts of cortical organization. *Brain*, *141*(11), 3262-3278. https://doi.org/10.1093/brain/awy273
Zimeo Morais, G. A., Balardin, J. B., & Sato, J. R. (2018). fNIRS Optodes' Location Decider (fOLD): a toolbox for probe arrangement guided by brain regions-of-interest. *Scientific Reports*, *8*(1).

https://doi.org/10.1038/s41598-018-21716-z

Appendix A: Code for Live Script

Pipeline for fNIRS connectivity analysis

Live script as a supplementary approach for GUI.

Created by Sindre Lilleseth 2022.

Initialization: Load data

```
 clear all;
 clc;
 close all;

 %%Choose folder for repository
 lib_folder = uigetdir([], 'Choose the folder containing the repository'); %
Choose the folder containing the library for this pipeline.
 addpath(genpath(lib_folder)); % Sets the library folder as a path.

 %%Load necessary data.
 load("BrainRegions_table.mat"); % Table with the different brain regions and
related data. Based on the table from OsloMet Phd candidate Haroon Khan.
 load("orderOfChan_mat.mat"); % The order of the channels based on brain
regions and optode placement.
 load("labelCells_comp.mat"); % The organised order of the regions for the
Circular Plot.

 %%Choose folder
 test_folder = uigetdir([], 'Choose folder containing the test data');
 subjsplit = split(test_folder, '\'); % Split folder path into cells. Based on
the hierarchy of this experiment data.

 %%Load nirs with nirs toolbox function
 addpath(genpath(test_folder)); % Adds the test folder to the path.
 nirsfile = strcat(subjsplit(length(subjsplit)),'.nirs'); % Find the name of
the nirs file.
 raw_nirs = nirs.io.loadDotNirs(nirsfile); % Loads the nirs file.

 %%Create pdf name
 pdf_name = cell2str(strcat(subjsplit(length(subjsplit)-1),'-
',subjsplit(length(subjsplit)),'.pdf'));

 %%Choose folder for storing the report
 [filename_report, filepath_report] = uiputfile(pdf_name,'Choose folder for
storing the report');

 %%Create folder for storing the report and figures.
 folder_name = cell2str(strcat(subjsplit(length(subjsplit)-1),'-
',subjsplit(length(subjsplit))));
 new_folder = strcat(filepath_report,folder_name);
```

```
mkdir(new_folder);


%%Defining some variables:
satlength = 2; % The saturation length of the bad channel rejection
algorithm.
QCoDthresh = 0.1; % The Quartile Coeffisient of Dispersion threshold used in
the bad channel rejection function.
```

Step 1: Rename stimuli data.

```
job1 = nirs.modules.RenameStims; % Define the first job: Rename stimuli.
job1.listOfChanges = {'stim_channel1' 'Start/stop'}; % Rename the stimuli.
raw_nirs_ReStim = job1.run(raw_nirs); % Perform job1 on the raw data.

figure(1); % Define a figure.
raw_nirs_ReStim.draw % Plot the figure.
xlabel('Seconds'), ylabel('Amplitude');
title('Raw data w/stimuli');

ax1 = gca; % Get the properties of the axes.
exportgraphics(ax1, [filepath_report filename_report], 'Resolution', 500); %
Export the plot to the report.
exportgraphics(ax1, fullfile(new_folder, strcat(folder_name,'-',
'Raw_stimuli','.png')),"Resolution",500);

%StatusEditField.Value = 'Data is renamed'; % Updating the status field.
close(figure(1));
```

Step 2: Trim data

```

job2 = nirs.modules.TrimBaseline; % Defining the second job: Trimming of the
data.
job2.preBaseline = -5; % Removing the first 5 seconds of the data.
raw_nirs_Trimmed = job2.run(raw_nirs_ReStim); % Performing job2 on the ReStim
data.

job2.postBaseline = -10; % Removing the last 10 seconds of the data.
raw_nirs_Trimmed = job2.run(raw_nirs_Trimmed); % Performing the updated job2
on the data.

figure(2); % Defining a new figure.
raw_nirs_Trimmed.draw; % Plotting the trimmed data.
xlabel('Seconds'), ylabel('Amplitude');
title('Trimmed raw data');

ax2 = gca; % Get the properties of the axes.
```

```
 exportgraphics(ax2, [filepath_report filename_report], 'Resolution', 500,
'append', true); % Export the plot to the report.
 exportgraphics(ax2, fullfile(new_folder, strcat(folder_name,'-
','Raw_trimmed','.png')),"Resolution",500);
 close(figure(2));
```

Step 3: Label short separation channels

```
 job3 = nirs.modules.LabelShortSeperation(job2); % Defining the third job:
Label the short separation channels.
 job3.max_distance = 15; % Setting the max source detector distance to 15 mm.
 raw_nirs_Labeled = job2.run(raw_nirs_Trimmed); % Perform job 3 on the trimmed
data.
```

Step 4: Correct baseline

```
 StatusEditField.Value = 'Correcting baseline...'; % Updating the status
field.
 job4 = nirs.modules.BaselineCorrection(job3); % Defining the fourth job:
Correction of the baseline.
 Baseline_corrected = job4.run(raw_nirs_Labeled); % Perform job4 on the
labeled data.

 figure(4);
 Baseline_corrected.draw; % Plotting of the baseline corrected data.
 xlabel('Seconds'), ylabel('Amplitude');
 title('Baseline corrected raw data');

 ax4 = gca; % Get the properties of the axes.
 exportgraphics(ax4, [filepath_report filename_report], 'Resolution', 500,
'append', true); % Exporting the plot to the report.
 exportgraphics(ax4, fullfile(new_folder, strcat(folder_name,'-
','Raw_baselineCorr','.png')),"Resolution",500); % Exporting the plot as a
single file.
 close(figure(4));
```

Step 5: Remove bad channels

```
 % Finding the bad channels in the data.
 [temp_data, channelmask] =
removeBadChannels_modified(Baseline_corrected.data, Baseline_corrected.Fs,
satlength, QCoDthresh);

 bad_chans_str_temparr = strings(1,length(channelmask)); %Defining a temporary
string array.
 bad_chans_count = 1;

 for i=1: length(channelmask)
     % Removing bad channels from the data.
     if channelmask(1,i) == 0
         Baseline_corrected.data(:,i) = 0;
         bad_chans_str_temparr(1,bad_chans_count) = i; bad_chans_count = 
bad_chans_count + 1; % Store the bad channel number for reporting.
     end
 end

 if bad_chans_count == 1
     bad_chans_str = 'No bad channels detected';

 else
     % Create string of the bad channels detected.
     bad_chans_str_arr = strings(1,bad_chans_count-1);
     for k=1: bad_chans_count-1
         bad_chans_str_arr(1,k) = bad_chans_str_temparr(1,k);
     end
     bad_chans_str = strjoin(bad_chans_str_arr,', ');
 end
```

Step 6: Removing short channels

```
 noshortchans = Baseline_corrected; % Creates a copy of the baseline corrected 
data.
 rangeOfLoop = length(Baseline_corrected.probe.link.ShortSeperation); % 
Creates an integer for the removal loop.

 for i=1: rangeOfLoop
     % For-loop for removing the short separation optodes from
     % the data
     if Baseline_corrected.probe.link.ShortSeperation(rangeOfLoop-i+1) == 1
         noshortchans.data(:,rangeOfLoop-i+1) = 0;
     end
 end

 figure(6);
 noshortchans.draw; % Plotting of the data without short channels.
 xlabel('Seconds'), ylabel('Amplitude');
 title('Removed short and bad channels');

 ax6 = gca; % Get the properties of the axes.
 exportgraphics(ax6, [filepath_report filename_report], 'Resolution', 500, 
'append', true); % Exporting the plot to the report.
 exportgraphics(ax6, fullfile(new_folder, strcat(folder_name,'-
','Channels_removed','.png')),"Resolution",500); % exporting the plot as 
single file.
 close(figure(6));
```

Step 7: Intensity to Optical Density

```
 job7 = nirs.modules.OpticalDensity(job4); % Defining job 7. Includes job4 to
log the process.
 OD = job7.run(noshortchans); % Perform job7 on the data.

 for i=1:length(channelmask)
     % Necessary step to ensure the correct format of the data
     % after this pre-processing step as the short channels are set to NaN.
     if channelmask(1,i) == 0
         OD.data(:,i) = 0;
     end
 end

 figure(7);
 OD.draw; % Plots the OD data.
 xlabel('Seconds'), ylabel('OD');
 title('Optical Density');

 ax7 = gca;
 exportgraphics(ax7, [filepath_report filename_report], 'Resolution', 500,
'append', true); % Exporting the plot to report.
 exportgraphics(ax7, fullfile(new_folder, strcat(folder_name,'-
','OD','.png')),"Resolution",500); % Exporting the plot as single file.
 close(figure(7));
```

Step 8: Optical density to hbo and hbr concentrations

```
 job8 = nirs.modules.BeerLambertLaw(); % By default: PPF = 0.1. Used by
Santosa et al. 2018.
 hb = job8.run(OD); % Performing the mBLL on the OD data.

 for i=1:length(channelmask)
     %To prevent cells from being 'NaN'.
     if isnan(hb.data(1,i))
         hb.data(:,i) = 0;
     end
 end

 % Plot the figure
 figure(8);
 hb.draw;
 xlabel('Seconds'), ylabel('Conc');
 title('Concentrations');

 % Export plot.
 ax8 = gca;
 exportgraphics(ax8, [filepath_report filename_report], 'Resolution', 500,
'append', true);
 exportgraphics(ax8, fullfile(new_folder, strcat(folder_name,'-
','Conc','.png')),"Resolution",500);
```

```
 close(figure(8));
```

Step 9: Bandpass filtering

```
 hb_filter_job = eeg.modules.BandPassFilter(); % Applying a bandpass filter to
the process.

 hb_filter_job.do_downsample = false; %Do not downsample data
 hb_filter_job.lowpass = 0.5; % Setting the cutoff frequency for the lowpass
filter = 0.5 Hz. Based on Yucel et al. 2021.
 hb_filter_job.highpass = 0.01; % Setting the cutoff frequency for the
highpass filter = 0.01 Hz. Based on Yucel et al. 2021.

 hb_filtered = hb_filter_job.run(hb); % Execute the filter job on the
concentration data.

 % Plot figure
 figure(9);
 hb_filtered.draw;
 xlabel('Seconds'), ylabel('Conc');
 title('Concentrations - filtered');

 % Export data.
 ax9 = gca;
 exportgraphics(ax9, [filepath_report filename_report], 'Resolution', 500,
'append', true);
 exportgraphics(ax9, fullfile(new_folder, strcat(folder_name,'-
','Conc_filtered','.png')),"Resolution",500);
 close(figure(9));
```

Step 10: Reorganize the data based on the brain regions.

```
 reOrg_data =
zeros(numel(hb_filtered.data(:,1)),numel(hb_filtered.data(1,:))); % Defining a
new matrix with the dimensions of the filtered data.

 for i=1: numel(hb_filtered.data(1,:))
     % Reorganizing the data.
     reOrg_data(:,i) = hb_filtered.data(:,orderOfChan_mat(i));
 end
```

Step 11: Delete the short seperation channels

```
 %%Deletes the short channels from the reorganized data
 for i=18:49 % i=18: 49 is based on the removal of the channels starting from
the last column.
     reOrg_data(:,rangeOfLoop-i) = [];
     %Reorganized data has the alphabetic order of the brain region table.
     %F.ex: Column 1 = Channel 39 (hbo channel of the "Area between
     %brodmann 42L and 02L  Ant. & posterior transverse temporal and
     %Primary Somatosensory Cortex").
     %This needs to be changed when
 end
```

Step 12: GLM

```
jobs_GLM = nirs.modules.Resample();
%app.jobs_GLM = nirs.modules.RenameStims(app.jobs_GLM); % Experience unknown
error
%in this step
%app.jobs_GLM.listOfChanges = {'stim_channel1' 'Start/stop' 'stim_channel1'
%'Start/stop'}; % Rename the stimuli. % Experience unknown error in this
%step.
jobs_GLM = nirs.modules.OpticalDensity(jobs_GLM);
jobs_GLM = nirs.modules.BeerLambertLaw(jobs_GLM);

hb_GLM=jobs_GLM.run(raw_nirs);

jobs_GLM_hb=nirs.modules.GLM(); %Starts a new group of jobs.
jobs_GLM_hb=nirs.modules.ExportData(jobs_GLM_hb);
jobs_GLM_hb.Output="GLM_analysis";
data_GLM=jobs_GLM_hb.run(hb_GLM);

data_GLM.draw('tstat', [-10 10], 'p<0.05');

ax12a = gca(1);
ax12b = gca(2);

%app.StatusEditField.Value = 'GLM completed'; % Updating the status field.

exportgraphics(ax12a, [filepath_report filename_report], 'Resolution', 500,
'append', true);
exportgraphics(ax12a, fullfile(new_folder, strcat(folder_name,'-
','GLM_1','.png')),"Resolution",500);
exportgraphics(ax12b, [filepath_report filename_report], 'Resolution', 500,
'append', true);
exportgraphics(ax12b, fullfile(new_folder, strcat(folder_name,'-
','GLM_2','.png')),"Resolution",500);

close all;
```

Step 13: Functional Connectivity

```
[R_vals_ar, P_vals_ar, dfe_ar] = nirs.sFC.ar_corr(reOrg_data);

threshold = [0.5:0.05:1]; % The thresholds for the connectivity matrix.
Increases by 5% each plot.
```

```
 conn_mat = R_vals_ar; %Copy the R-values (correlation values) matrix. Change
R-values matrix depending on utilized FC function.

 figure(13);
 imagesc(conn_mat); title('Connectivity matrix'); xlabel('Channels');
ylabel('Channels');
 ax13a = gca; colorbar; %app.ax13a.Name = "Connectivity matrix";
 exportgraphics(ax13a, [filepath_report filename_report], 'Resolution', 500,
'append', true);
 exportgraphics(ax13a, fullfile(new_folder, strcat(folder_name,'-
','Connectivity_matrix','.png')),"Resolution",500);
 close(figure(13)); pause(1);

 %%Creating connectivity matrices with thresholded values.
 for t=1:length(threshold)
     thresholded_connmat = zeros(length(conn_mat),length(conn_mat)); %%Create
an empty matrix.

     for i=1:length(conn_mat)
         for j=1:length(conn_mat)
             if conn_mat(i,j) > threshold(t)
                 thresholded_connmat(i,j) = conn_mat(i,j);
             end
         end
     end


     % Plots and exports the thresholded matrix.
     mat_title_2 = sprintf('Thresholded matrix. Threshold = %d',
threshold(t));
     figure(14);
     imagesc(thresholded_connmat); title(mat_title_2); xlabel('Channels');
ylabel('Channels');
     ax13b = gca; colorbar; %app.ax13b.Name = "Connectivity matrix
(thresholded)";
     exportgraphics(ax13b, [filepath_report filename_report], 'Resolution',
500, 'append', true);
     exportgraphics(ax13b, fullfile(new_folder, strcat(folder_name,'-
','Thresholded_ConnMat_',num2str(t),'.png')),"Resolution",500);
     close(figure(14)); pause(1);


     % Plots and exports the circular graph.
     figure(15);
     CircGraphName = sprintf('Circular graph. Threshold = %d',threshold(t));
     circularGraph(thresholded_connmat,'Label',labelCells(2,:)); %Input: The
adjacency matrix.
     ax13c = gcf;
     ax13c.Name = CircGraphName;
     ax13c.Position = [200 200 1000 1000]; %Changes the size of the plot to
ensure space between labels.
     exportgraphics(ax13c, [filepath_report filename_report], 'Resolution',
500, 'append', true);
     exportgraphics(ax13c, fullfile(new_folder, strcat(folder_name,'-
','Thresholded_CircGraph_',num2str(threshold(t)),'.png')),"Resolution",500);
     close(figure(15)); pause(1);
 end

 pause(3);
```

Step 14: Save all workspace variables in report folder and relevant information.

```
 save(strcat(new_folder,'\', folder_name,'.mat'));  % Save the workspace
variable to the subject folder.

 % The bad channels detected.
 str_bad_chans = strcat(['Bad channels: ', bad_chans_str]);

 % Add bad channel information to the report.
 figure(18);
 annotation('textbox', [.2 .3 .7 .7], 'String', str_bad_chans,'FontSize', 8);
 ax18 = gcf;
 exportgraphics(ax18, [filepath_report filename_report], 'Resolution', 500,
'append', true);
 close(figure(18));

 % The acronyms for the Brodmann areas.
 str_BA = sprintf(['Description of acronyms for the Brodmann Areas: \n' ...
     'o: Oxygenated channel\n' ...
     'r: Deoxygenated channel\n\n' ...
     '42L/02L:    Area between brodmann 42L and 02L Ant. & posterior
transverse temporal and Primary Somatosensory Cortex (Secondary auditory
cortex)\n' ...
     '41R/01R:   Area between 41R and 01R Ant. & posterior transverse temporal
and primary somatosensory cortex\n' ...
     'MTG:   Middle Temporal Gyrus\n' ...
     'MTSG:  Middle Temporal Gyrus and Superior Temporal Gyrus\n' ...
     'PTB:   Pars Triangularis Brocas Area\n' ...
     'PMS:   Pre-motor and Supplementary Motor Cortex\n' ...
     'PFC:   Prefrontal Cortex\n' ...
     'PMC:   Primary Motor cortex\n' ...
     'RSA:   Retrosubicular Area\n' ...
     'SCA:   Subcentral Area\n' ...
     'STG:   Superior Temporal Gyrus\n' ...
     'SGWA:  Supramarginal Gyrus of Wernickes Area\n' ...
     '\n']);

 % Add abbrevations to the report.
 figure(16);
 annotation('textbox', [.2 .3 .7 .7], 'String', str_BA,'FontSize', 8);
 ax16 = gcf;
 exportgraphics(ax16, [filepath_report filename_report], 'Resolution', 500,
'append', true);
 close(figure(16));

 % The summary of the processing.
 str_sum = sprintf(['The variables and functions used in this pipeline: \n'...
     'Partial Path Length Factor: 0.1 (default in nirs toolbox).\n'...
     'Quartile Coeffisient of Dispersion: %s.\n'... %add QCoD as first
variable.
     'Saturation length (bad channel rejection): %ds.\n'...
     '\n' ...
     'Filter: 4th order butterworth bandpass filter.\n' ...
     'Lowpass cutoff: %dHz\n' ...
     'Highpass cutoff: %dHz\n' ...
     '\n' ...
     'FC function: Positive correlation prewhitened with an autoregressive
(AR) model.\n' ...
     'FC measure: Pearsons R.\n' ...
     '\n' ...
     'General Linear Model: Autoregressive model based algorithm for
correcting motion and serially correlated errors in fNIRS by Barker et al.
(2013)'],num2str(QCoDthresh), satlength, hb_filter_job.lowpass,
hb_filter_job.highpass);

 % Add summary to the report.
 figure(17);
 annotation('textbox', [.2 .3 .7 .7], 'String', str_sum, 'FontSize', 8);
 ax17 = gcf;
 exportgraphics(ax17, [filepath_report filename_report], 'Resolution', 500,
'append', true);
 close(figure(17));

 close all;
```

Appendix B: Code for GUI in MATLAB App designer

```
classdef fNIRS_Connectivity_GUI_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        fNIRSConnectivityGUIOsloMetUIFigure  matlab.ui.Figure
        PipelineforfNIRSconnectivityanalysisv1Label  matlab.ui.control.Label
        Image                 matlab.ui.control.Image
        StatusEditField       matlab.ui.control.EditField
        StatusEditFieldLabel  matlab.ui.control.Label
        RunpipelineButton     matlab.ui.control.Button
    end


    properties (Access = private)
        %%Defining all variables and objects.

         %%Loading data:
         lib_folder; % Folder containing all necessary functions.
         test_folder; % Folder containing the subject data.
         subjsplit; % Cells containing the folder path in splitted format.
         raw_nirs; % The raw nirs data.
         nirsfile; % Name of the nirs loaded nirs file.
         pdf_name; % Name of the created pdf report for the current subject.
         filename_report; % File name of the report.
         filepath_report; % Path of the report.
         folder_name; % The name of the new folder.
         new_folder; % The path of the new folder.


         %%Step 1: Rename stimuli
         job1; % Definition of the job: RenameStims.
         raw_nirs_ReStim; % The data with new stimuli name.
         ax1; % The axes information of the rename stims plot.

         %%Step 2: Trimming data
         job2; % Definition of the job: TrimBaseline.
         raw_nirs_Trimmed; % The trimmed data.
         ax2; % The axes information of the TrimBaseline plot.

         %%Step 3: Label short separation channels
         job3; % Definition of the job: LabelShortSeparation.
         raw_nirs_Labeled; % The labeled data.

         %%Step 4: Correct baseline
         job4; % Definition of the job: BaselineCorrection.
         Baseline_corrected; % The baseline corrected data.
         ax4; %The axes information of the Baseline plot.

         %%Step 5: Remove bad channels
         QCoDthresh = 0.1; % Defining the threshold value of the Quartile Coefissient of Dispersion.
         satlength = 2; % Defining the saturation length. 2 s by default.
         temp_data; % Temporary storage of unneccesary data.
         channelmask; % The matrix labeling bad channels.
         bad_chans_str; % String for bad channels

         %%Step 6: Remove short channels
         noshortchans; % The data without the short channels
         rangeOfLoop; % Variable for data removal.
         ax6; % The axes information of the data without short and bad channels.

         %%Step 7: Intensity to Optical Density
         job7; % Defintion of the job: OpticalDensity
         OD; % The optical density data.
         ax7; % The axes information of the OD data.

         %%Step 8: Optical denisity to Concentrations
         job8; % Definition of the job: BeerLambertLaw.
         hb; % The concentration data.
         ax8; % The axes informtation of the concentration data.

         %%Step 9: Bandpass filtering
         hb_filter_job; % Definition of the filtering job.
         hb_filtered; % The filtered data.
         ax9; % The axes information of the filtered plot.

         %%Step 10: Reorganizing the data
         reOrg_data; % The reorganized data.

         %%Step 12: GLM
         jobs_GLM; % Defining the GLM jobs.
         hb_GLM; % The concentration data before the GLM.
         jobs_GLM_hb; % Defining the second part of jobs.
         data_GLM; % The data from the GLM processing.
         ax12a; % The axes information of the GLM hbo
         ax12b; % The axes information of the GLM hbr

         %%Step 13: Functional Connectivity
         R_vals_ar; % R-values from the AR model.
         P_vals_ar; % P-values from the AR model.
         dfe_ar; % The degrees of freedom from the AR model.
         threshold; % The threshold value given by the user.
         conn_mat; % The connectivity matrix
         thresholded_connmat; % The thresholded connectivity matrix.
         bin_mat; % The binarized matrix
         ax13a;
         ax13b;
         ax13c;
     end

     % Callbacks that handle component events
     methods (Access = private)

         % Button pushed function: RunpipelineButton
         function RunpipelineButtonPushed(app, event)
             %% Initialization: Load data
             %%Choose folder for repository
             app.lib_folder = uigetdir([], 'Choose the folder containing the repository'); % Choose the folder containing the library for this
 pipeline.
             addpath(genpath(app.lib_folder)); % Sets the library folder as a path.

             %%Load necessary data.
             load("BrainRegions_table.mat"); % Table with the different brain regions and related data. Based on the table from H. Khan.
             load("orderOfChan_mat.mat"); % The order of the channels based on brain regions and optode placement.
             load("labelCells_comp.mat"); % The organised order of the regions for the Circular Plot.

             %%Choose folder
             app.test_folder = uigetdir([], 'Choose folder containing the test data');
             app.subjsplit = split(app.test_folder, '\'); % Split folder path into cells. Based on the hierarchy of this experiment data.

             %%Load nirs with nirs toolbox function
             addpath(genpath(app.test_folder)); % Adds the test folder to the path.
             app.nirsfile = strcat(app.subjsplit(length(app.subjsplit)),'.nirs'); % Find the name of the nirs file.
             app.raw_nirs = nirs.io.loadDotNirs(app.nirsfile); % Loads the nirs file.

             %%Create pdf name
             app.pdf_name = cell2str(strcat(app.subjsplit(length(app.subjsplit)-1),'-',app.subjsplit(length(app.subjsplit)),'.pdf'));

             %%Choose folder for storing the report
             [app.filename_report, app.filepath_report] = uiputfile(app.pdf_name,'Choose folder for storing the report');

             %%Create folder for storing the report and figures.
             app.folder_name = cell2str(strcat(app.subjsplit(length(app.subjsplit)-1),'-',app.subjsplit(length(app.subjsplit))));
             app.new_folder = strcat(app.filepath_report,app.folder_name);
             mkdir(app.new_folder);

             app.StatusEditField.Value = 'Data is loaded'; % Updating the status field.
             %% Step 1: Rename stimuli data.
             app.job1 = nirs.modules.RenameStims; % Define the first job: Rename stimuli.
             app.job1.listOfChanges = {'stim_channel1' 'Start/stop'}; % Rename the stimuli.
             app.raw_nirs_ReStim = app.job1.run(app.raw_nirs); % Perform job1 on the raw data.

             figure(1); % Define a figure.
             app.raw_nirs_ReStim.draw % Plot the figure.
             xlabel('seconds'), ylabel('amplitude');
             title('Raw data w/stimuli');

             app.ax1 = gca; % Get the properties of the axes.
             exportgraphics(app.ax1, [app.filepath_report app.filename_report], 'Resolution', 500); % Export the plot to the report.
             exportgraphics(app.ax1, fullfile(app.new_folder, strcat(app.folder_name,'-', 'Raw_stimuli','.png')),"Resolution",500); % Export the
 plot to the folder.

             app.StatusEditField.Value = 'Data is renamed'; % Updating the status field.
             close(figure(1));



             %% Step 2: Trim data
             app.job2 = nirs.modules.TrimBaseline; % Defining the second job: Trimming of the data.
             app.job2.preBaseline = -5; % Removing the first 5 seconds of the data.
             app.raw_nirs_Trimmed = app.job2.run(app.raw_nirs_ReStim); % Performing job2 on the ReStim data.

             app.job2.postBaseline = -10; % Removing the last 10 seconds of the data.
             app.raw_nirs_Trimmed = app.job2.run(app.raw_nirs_Trimmed); % Performing the updated job2 on the data.

             figure(2); % Defining a new figure.
             app.raw_nirs_Trimmed.draw; % Plotting the trimmed data.
             xlabel('seconds'), ylabel('amplitude');
             title('Trimmed raw data');

             app.ax2 = gca; % Get the properties of the axes.
             exportgraphics(app.ax2, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true); % Export the plot to the
 report.
             exportgraphics(app.ax2, fullfile(app.new_folder, strcat(app.folder_name,'-','Raw_trimmed','.png')),"Resolution",500);

             app.StatusEditField.Value = 'Data is trimmed'; % Updating the status field.
             close(figure(2));

             %% Step 3: Label short separation channels
             app.job3 = nirs.modules.LabelShortSeperation(app.job2); % Defining the third job: Label the short separation channels.
             app.job3.max_distance = 15; % Setting the max source detector distance to 15 mm.
             app.raw_nirs_Labeled = app.job2.run(app.raw_nirs_Trimmed); % Perform job 3 on the trimmed data.

             app.StatusEditField.Value = 'Stimuli is labeled'; % Updating the status field.

             %% Step 4: Correct baseline
             app.StatusEditField.Value = 'Correcting baseline...'; % Updating the status field.
             app.job4 = nirs.modules.BaselineCorrection(app.job3); % Defining the fourth job: Correction of the baseline.
             app.Baseline_corrected = app.job4.run(app.raw_nirs_Labeled); % Perform job4 on the labeled data.

             figure(4);
             app.Baseline_corrected.draw; % Plotting of the baseline corrected data.
             xlabel('seconds'), ylabel('amplitude');
             title('Baseline corrected raw data');

             app.ax4 = gca; % Get the properties of the axes.
             exportgraphics(app.ax4, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true); % Exporting the plot to the
 report.
             exportgraphics(app.ax4, fullfile(app.new_folder, strcat(app.folder_name,'-','Raw_baselineCorr','.png')),"Resolution",500);

             app.StatusEditField.Value = 'Baseline is corrected'; % Updating the status field.
             close(figure(4));

             %% Step 5: Remove bad channels
             app.StatusEditField.Value = 'Removing bad channels...'; % Updating the status field.

             % Finding the bad channels in the data.
             [app.temp_data, app.channelmask] = removeBadChannels_modified(app.Baseline_corrected.data, app.Baseline_corrected.Fs,
 app.satlength, app.QCoDthresh);

             bad_chans_str_temparr = strings(1,length(app.channelmask));
             bad_chans_count = 1;

             for i=1: length(app.channelmask)
               % Removing bad channels from the data.
               if app.channelmask(1,i) == 0
                 app.Baseline_corrected.data(:,i) = 0;
                 bad_chans_str_temparr(1,bad_chans_count) = i; bad_chans_count = bad_chans_count + 1; % Store the bad channel number for
 reporting.
               end
             end

             if bad_chans_count == 1
               app.bad_chans_str = 'No bad channels detected';

             else
               bad_chans_str_arr = strings(1,bad_chans_count-1);
               for k=1: bad_chans_count-1
                 bad_chans_str_arr(1,k) = bad_chans_str_temparr(1,k);
               end
               app.bad_chans_str = strjoin(bad_chans_str_arr,', ');
             end

             app.StatusEditField.Value = 'Bad channels are removed'; % Updating the status field.

             %% Step 6: Removing short channels
             app.StatusEditField.Value = 'Removing short channels...'; % Updating the status field.

             app.noshortchans = app.Baseline_corrected; % Creates a copy of the baseline corrected data.
             app.rangeOfLoop = length(app.Baseline_corrected.probe.link.ShortSeperation); % Creates an integer for the removal loop.

             for i=1: app.rangeOfLoop
               % For loop for removing the short separation optodes from
               % the data
               if app.Baseline_corrected.probe.link.ShortSeperation(app.rangeOfLoop-i+1) == 1
                 app.noshortchans.data(:,app.rangeOfLoop-i+1) = 0;
               end
             end

             figure(6);
             app.noshortchans.draw; % Plotting of the data without short channels.
             xlabel('seconds'), ylabel('amplitude');
             title('Removed short and bad channels');

             app.ax6 = gca; % Get the properties of the axes.
             exportgraphics(app.ax6, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true); % Exporting the plot to the
 report.
             exportgraphics(app.ax6, fullfile(app.new_folder, strcat(app.folder_name,'-','Channels_removed','.png')),"Resolution",500);

             close(figure(6));
             app.StatusEditField.Value = 'Short channels are removed'; % Updating the status field.

             %% Step 7: Intensity to Optical Density
             app.StatusEditField.Value = 'Converting int2OD...'; % Updating the status field.
             app.job7 = nirs.modules.OpticalDensity(app.job4); % Defining job 7. Includes job4 to log the process.
             app.OD = app.job7.run(app.noshortchans); % Perform job7 on the data.

             for i=1:length(app.channelmask)
               % Necessary step to ensure the correct format of the data
               % after this pre-processing step.
               if app.channelmask(1,i) == 0
                 app.OD.data(:,i) = 0;
               end
             end

             figure(7);
             app.OD.draw; % Plots the OD data.
             xlabel('seconds'), ylabel('amplitude');
```

```
title('Optical Density');

app.ax7 = gca;
exportgraphics(app.ax7, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true); % Exporting the plot.
exportgraphics(app.ax7, fullfile(app.new_folder, strcat(app.folder_name,'-','OD','.png')),"Resolution",500);

close(figure(7));
app.StatusEditField.Value = 'Converted intensity to OD'; % Updating the status field.

%% Step 8: Optical density to hbo and hbr concentrations
app.StatusEditField.Value = 'Converting OD to conc...'; % Updating the status field.
app.job8 = nirs.modules.BeerLambertLaw(); % By default: PPF = 0.1. Used by Santosa et al. 2018.
app.hb = app.job8.run(app.OD); % Performing the mBLL on the OD data.

for i=1:length(app.channelmask)
   %To prevent cells from being 'NaN'.
   if isnan(app.hb.data(1,i))
      app.hb.data(:,i) = 0;
   end
end

figure(8);
app.hb.draw;
xlabel('seconds'), ylabel('amplitude');
title('Concentrations');

app.ax8 = gca;
exportgraphics(app.ax8, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
exportgraphics(app.ax8, fullfile(app.new_folder, strcat(app.folder_name,'-','Conc','.png')),"Resolution",500);

close(figure(8));
app.StatusEditField.Value = 'Converted OD to concentrations'; % Updating the status field.

%% Step 9: Bandpass filtering
app.StatusEditField.Value = 'Filter data...'; % Updating the status field.
app.hb_filter_job = eeg.modules.BandPassFilter(); % Applying a bandpass filter to the process.

app.hb_filter_job.do_downsample = false; %Do not downsample data
app.hb_filter_job.lowpass = 0.5; % Setting the cutoff frequency for the lowpass filter = 0.5 Hz.
app.hb_filter_job.highpass = 0.01; % Setting the cutoff frequency for the highpass filter = 0.01 Hz. Based on Yucel et al. 2021.

app.hb_filtered = app.hb_filter_job.run(app.hb); % Execute the filter job on the concentration data.

figure(9);
app.hb_filtered.draw;
xlabel('seconds'), ylabel('amplitude');
title('Concentrations - filtered');

app.ax9 = gca;
exportgraphics(app.ax9, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
exportgraphics(app.ax9, fullfile(app.new_folder, strcat(app.folder_name,'-','Conc_filtered','.png')),"Resolution",500);

close(figure(9));
app.StatusEditField.Value = 'Data is filtered'; % Updating the status field.

%% Step 10: Reorganize the data based on the brain regions.
app.reOrg_data = zeros(numel(app.hb_filtered.data(:,1)),numel(app.hb_filtered.data(1,:))); % Defining a new matrix with the
dimensions of the filtered data.

for i=1: numel(app.hb_filtered.data(1,:))
   % Reorganizing the data.
   app.reOrg_data(:,i) = app.hb_filtered.data(:,orderOfChan_mat(i));
end

app.StatusEditField.Value = 'Data is reorganized'; % Updating the status field.

%% Step 11: Delete the short seperation channels
%%Deletes the short channels from the reorganized data
for i=18:49 % i=18: 49 is based on the removal of the channels starting from the last column.
   app.reOrg_data(:,app.rangeOfLoop-i) = [];
   %Reorganized data has the alphabetic order of the brain region table.
```

```
        %F.ex: Column 1 = Channel 39 (hbo channel of the "Area between
        %brodmann 42L and 02L  Ant. & posterior transverse temporal and
        %Primary Somatosensory Cortex")
    end

    %% Step 12: GLM
    app.StatusEditField.Value = 'Running GLM...'; % Updating the status field.

    app.jobs_GLM = nirs.modules.Resample();
    %app.jobs_GLM = nirs.modules.RenameStims(app.jobs_GLM);
    %app.jobs_GLM.listOfChanges = {'stim_channel1' 'Start/stop' 'stim_channel1' 'Start/stop'}; % Rename the stimuli.
    app.jobs_GLM = nirs.modules.OpticalDensity(app.jobs_GLM);
    app.jobs_GLM = nirs.modules.BeerLambertLaw(app.jobs_GLM);

    app.hb_GLM=app.jobs_GLM.run(app.raw_nirs);

    app.jobs_GLM_hb=nirs.modules.GLM(); %Starts a new group of jobs.
    app.jobs_GLM_hb=nirs.modules.ExportData(app.jobs_GLM_hb);
    app.jobs_GLM_hb.Output="GLM_analysis";
    app.data_GLM=app.jobs_GLM_hb.run(app.hb_GLM);

    app.data_GLM.draw('tstat', [-10 10], 'p<0.05');

    app.ax12a = gca(1);
    app.ax12b = gca(2);

    app.StatusEditField.Value = 'GLM completed'; % Updating the status field.

    exportgraphics(app.ax12a, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
    exportgraphics(app.ax12a, fullfile(app.new_folder, strcat(app.folder_name,'-','GLM_1','.png')),"Resolution",500);
    exportgraphics(app.ax12b, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
    exportgraphics(app.ax12b, fullfile(app.new_folder, strcat(app.folder_name,'-','GLM_2','.png')),"Resolution",500);

    close all;

    %% Step 13: Functional Connectivity
    app.StatusEditField.Value = 'Calculating FC...'; % Updating the status field.

    [app.R_vals_ar, app.P_vals_ar, app.dfe_ar] = nirs.sFC.ar_corr(app.reOrg_data);

%           prompt = {'Enter threshold for Functional Connectivity'};
%           dlgtitle = 'Threshold';
%           dims = [1];
%           definput = {'0.6'};
%           app.threshold = str2double(inputdlg(prompt,dlgtitle,dims,definput));

    app.threshold = [0.5:0.05:1]; % The thresholds for the connectivity matrix. Increases by 5% each plot.

    app.conn_mat = app.R_vals_ar; %Copy the R-values (correlation values) matrix. Change R-values matrix depending on utilized FC
function.

    figure(13);
    imagesc(app.conn_mat); title('Connectivity matrix');
    app.ax13a = gca; %app.ax13a.Name = "Connectivity matrix";
    exportgraphics(app.ax13a, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
    exportgraphics(app.ax13a, fullfile(app.new_folder, strcat(app.folder_name,'-','Connectivity_matrix','.png')),"Resolution",500);
    close(figure(13)); pause(1);

    %%Creating connectivity matrices with thresholded values.
    for t=1:length(app.threshold)
        app.thresholded_connmat = zeros(length(app.conn_mat),length(app.conn_mat)); %%Create an empty matrix.

        for i=1:length(app.conn_mat)
            for j=1:length(app.conn_mat)
                if app.conn_mat(i,j) > app.threshold(t)
                    app.thresholded_connmat(i,j) = app.conn_mat(i,j);
                end
            end
        end

        mat_title_2 = sprintf('Thresholded matrix. Threshold = %d', app.threshold(t));
```

```
            figure(14);
            imagesc(app.thresholded_connmat); title(mat_title_2);
            app.ax13b = gca; %app.ax13b.Name = "Connectivity matrix (thresholded)";
            exportgraphics(app.ax13b, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
            exportgraphics(app.ax13b, fullfile(app.new_folder, strcat(app.folder_name,'-
','Thresholded_ConnMat_',num2str(t),'.png')),"Resolution",500);
            close(figure(14)); pause(1);



            figure(15);
            CircGraphName = sprintf('Circular graph. Threshold = %d',app.threshold(t));
            circularGraph(app.thresholded_connmat,'Label',labelCells(2,:)); %Input: The adjacency matrix.
            app.ax13c = gcf;
            app.ax13c.Name = CircGraphName;
            app.ax13c.Position = [200 200 1000 1000]; %Changes the size of the plot to ensure space between labels.
            exportgraphics(app.ax13c, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
            exportgraphics(app.ax13c, fullfile(app.new_folder, strcat(app.folder_name,'-
','Thresholded_CircGraph_',num2str(app.threshold(t)),'.png')),"Resolution",500);

            close(figure(15)); pause(1);
        end

        pause(1);
        app.StatusEditField.Value = 'Report is completed'; % Updating the status field.


        %% Step 14: Save all workspace variables in report folder.
        save(strcat(app.new_folder,'\', app.folder_name,'.mat'));

        % The bad channels detected.
        str_bad_chans = strcat(['Bad channels: ', app.bad_chans_str]);

        % Add bad channel information to the report.
        figure(18);
        annotation('textbox', [.2 .3 .7 .7], 'String', str_bad_chans,'FontSize', 8);
        ax18 = gcf;
        exportgraphics(ax18, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
        close(figure(18));

        % The acronyms for the Brodmann areas.
        str_BA = sprintf(['Description of acronyms for the Brodmann Areas: \n' ...
            'o: Oxygenated channel\n' ...
            'r: Deoxygenated channel\n\n' ...
            '42L/02L:    Area between brodmann 42L and 02L Ant. & posterior transverse temporal and Primary Somatosensory Cortex
(Secondary auditory cortex)\n' ...
            '41R/01R:    Area between 41R and 01R Ant. & posterior transverse temporal and primary somatosensory cortex\n' ...
            'MTG:   Middle Temporal Gyrus\n' ...
            'MTSG:  Middle Temporal Gyrus and Superior Temporal Gyrus\n' ...
            'PTB:   Pars Triangularis Brocas Area\n' ...
            'PMS:   Pre-motor and Supplementary Motor Cortex\n' ...
            'PFC:   Prefrontal Cortex\n' ...
            'PMC:   Primary Motor cortex\n' ...
            'RSA:   Retrosubicular Area\n' ...
            'SCA:   Subcentral Area\n' ...
            'STG:   Superior Temporal Gyrus\n' ...
            'SGWA:  Supramarginal Gyrus of Wernickes Area\n' ...
            '\n']);

        % Add summary to the report.
        figure(16);
        annotation('textbox', [.2 .3 .7 .7], 'String', str_BA,'FontSize', 8);
        ax16 = gcf;
        exportgraphics(ax16, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);
        close(figure(16));


        % The summary of the processing.
        str_sum = sprintf(['The variables and functions used in this pipeline: \n'...
            'Partial Path Length Factor: 0.1 (default in nirs toolbox).\n'...
            'Quartile Coeffisient of Dispersion: %s.\n'...
            'Saturation length (bad channel rejection): %ds.\n'...
            '\n' ...
            'Filtertype: 4th order butterworth bandpass filter.\n' ...
            'Lowpass cutoff: %d Hz\n' ...
            'Highpass cutoff: %d Hz\n' ...
            '\n' ...
            'FC function: Correlation prewhitened with an autoregressive (AR) model.\n' ...
            'FC measure: Pearsons R.\n'],num2str(app.QCoDthresh), app.satlength, app.hb_filter_job.lowpass, app.hb_filter_job.highpass);


        % Add summary to the report.
        figure(17);
        annotation('textbox', [.2 .3 .7 .7], 'String', str_sum, 'FontSize', 8);
        ax17 = gcf;
        exportgraphics(ax17, [app.filepath_report app.filename_report], 'Resolution', 500, 'append', true);

        app.StatusEditField.Value = 'Variables are stored in report folder'; % Updating the status field.
        close(figure(17));

        app.StatusEditField.Value = 'Pipeline completed. Ready for another run.'; % Updating the status field.

    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create fNIRSConnectivityGUIOsloMetUIFigure and hide until all components are created
        app.fNIRSConnectivityGUIOsloMetUIFigure = uifigure('Visible', 'off');
        app.fNIRSConnectivityGUIOsloMetUIFigure.Position = [100 100 640 480];
        app.fNIRSConnectivityGUIOsloMetUIFigure.Name = 'fNIRS Connectivity GUI - OsloMet';

        % Create RunpipelineButton
        app.RunpipelineButton = uibutton(app.fNIRSConnectivityGUIOsloMetUIFigure, 'push');
        app.RunpipelineButton.ButtonPushedFcn = createCallbackFcn(app, @RunpipelineButtonPushed, true);
        app.RunpipelineButton.Position = [62 330 100 22];
        app.RunpipelineButton.Text = 'Run pipeline';

        % Create StatusEditFieldLabel
        app.StatusEditFieldLabel = uilabel(app.fNIRSConnectivityGUIOsloMetUIFigure);
        app.StatusEditFieldLabel.HorizontalAlignment = 'right';
        app.StatusEditFieldLabel.Position = [236 330 43 22];
        app.StatusEditFieldLabel.Text = 'Status:';

        % Create StatusEditField
        app.StatusEditField = uieditfield(app.fNIRSConnectivityGUIOsloMetUIFigure, 'text');
        app.StatusEditField.Position = [294 330 252 22];

        % Create Image
        app.Image = uiimage(app.fNIRSConnectivityGUIOsloMetUIFigure);
        app.Image.Position = [2 381 100 100];
        app.Image.ImageSource = 'OsloMet logo for nett.png';

        % Create PipelineforfNIRSconnectivityanalysisv1Label
        app.PipelineforfNIRSconnectivityanalysisv1Label = uilabel(app.fNIRSConnectivityGUIOsloMetUIFigure);
        app.PipelineforfNIRSconnectivityanalysisv1Label.FontSize = 18;
        app.PipelineforfNIRSconnectivityanalysisv1Label.FontWeight = 'bold';
        app.PipelineforfNIRSconnectivityanalysisv1Label.Position = [129 419 390 23];
        app.PipelineforfNIRSconnectivityanalysisv1Label.Text = 'Pipeline for fNIRS connectivity analysis - v.1';

        % Show the figure after all components are created
        app.fNIRSConnectivityGUIOsloMetUIFigure.Visible = 'on';
    end
end

% App creation and deletion
methods (Access = public)
```

```
        % Construct app
        function app = fNIRS_Connectivity_GUI_exported

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.fNIRSConnectivityGUIOsloMetUIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.fNIRSConnectivityGUIOsloMetUIFigure)
        end
    end
end
```

Additional Appendices: Included in zip folder

Appendix C: The REK application nr: 322236

Appendix D: Repository for connectivity pipeline

Appendix E: Example of report