NeuroLight Lab: fNIRS Data Analysis Pipeline

Meram Mahmoud
Department of Biomedical Engineering,
Cairo University, Class of 2026

***Abstract*—This document describes the NeuroLight Lab project, a complete pipeline for analyzing functional Near-Infrared Spectroscopy (fNIRS) data. The pipeline includes data exploration, preprocessing, motion correction, and statistical modeling. It also provides a practical guide to handling fNIRS data with detailed examples for researchers and students.**

***Index Terms*—fNIRS, Data Analysis, Motion Correction, Statistical Models, Block Averaging, GLM, LMM**

I. Introduction

Welcome to **NeuroLight Lab**, a comprehensive project that showcases a full pipeline for analyzing functional Near-Infrared Spectroscopy (fNIRS) data. This project is designed for researchers and students working with brain imaging data and provides a hands-on walkthrough from raw data to statistical modeling and interpretation.

II. Data Description

The data used in this study comes from the dataset available at Mendeley Data. The associated study is titled "Examining the hemodynamic response during perception of noise-vocoded speech and speech in background noise: An image-based fNIRS study".

A. Notes on the Data

The dataset includes files named from S1001 to S1040, corresponding to 40 participants. However, no data files are available for subjects 2 and 15, meaning the dataset consists of data from 38 participants, which still exceeds the 31 participants mentioned in the paper.

The NIRS headband was developed to secure two light sources and four light detectors over each hemisphere, resulting in six measurable channels (30 mm in length) per hemisphere. As a result, we expect 12 channels in total (6 channels per hemisphere), with two wavelengths corresponding to HbO and HbR, giving a total of 24 channels. However, the dataset contains 28 NIRS channels, which is inconsistent with the manuscript description.

B. Trial Structure

The experiment used an event-related design, where stimuli were presented in blocks (instead of randomization). The conditions were randomized across subjects to avoid order effects. Each block began with three practice sentences for familiarization, followed by 20 randomized sentences. In total, there were 69 trials across all three conditions.

The trial structure was as follows:

- 500 ms silence
- 3000 ms sentence presentation
- 500-2000 ms silence
- A short audible click, followed by 3000 ms within which the participant repeats the sentence
- 1000-2000 ms silence

The total trial duration was between 8 and 10.5 seconds. This trial structure was consistent with the shared data, though there are some differences from the setup described in the paper.

C. Note on Preprocessing

The following steps demonstrate the typical preprocessing techniques for fNIRS data. These methods do not adhere to the exact pipeline used in the original study, but we use the dataset to perform our own preprocessing experiments.

III. Project Overview

This project demonstrates how to:

- Load and explore `.nirs` files
- Preprocess raw fNIRS signals
- Correct motion artifacts
- Apply robust statistical models, including:
  - Block Averaging
  - Generalized Linear Models (GLMs)
  - Linear Mixed Models (LMMs)

Whether you're just getting started or refining your own fNIRS analysis pipeline, this project bridges the gap between theory and practice with detailed examples.

IV. fNIRS Data Exploration

This section walks you through the process of exploring fNIRS data, including understanding the data structure, discovering the source-detector layout, calculating distances between optodes, and visualizing the raw signals.

A. Load the Data

To begin, load the `.nirs` file using the `matfile` function in MATLAB. Once the file is loaded, check the structure to understand its contents.

The file will contain several variables, including:

- `d`: Raw light intensity (channels × time)
- `t`: Time vector (in seconds)
- `s`: Stimulus vector (1 = event on)
- `aux`: Auxiliary signals (e.g., accelerometer data)
- `ml`: Measurement list (channel & wavelength mapping)
- `SD`: Source-detector geometry (layout and wavelengths)
- `systemInfo`: Metadata about the acquisition device

Name	Size	Bytes	Class	Attributes
SD	1x1	6773	struct	
aux	54675x8	3499200	double	
d	54675x28	12247200	double	
dStd	0x0	0	double	
ml	28x4	896	double	
s	54675x1	437400	double	
systemInfo	1x1	876818	struct	
t	54675x1	437400	double	
tdml	28x54675	12247200	double	

Fig. 1: file variables

For preprocessing, the main variables of interest are `d`, `t`, `s`, and `aux` for signals and events, and `SD` and `ml` for channel and optode information.

B. *Discover Optode Placement and Wavelengths*

To explore the source-detector layout and wavelengths, use the following MATLAB code (see images for visualizations):

1×2 double

	1	2
1	690	830

Fig. 2: wavelengths used

ml × wavelengths ×

28×4 double

	1	2	3	4
1	1	1	1	1
2	1	2	1	1
3	2	1	1	1
4	2	2	1	1
5	2	3	1	1
6	2	5	1	1
7	3	2	1	1
8	3	4	1	1
9	3	5	1	1
10	3	6	1	1
11	4	5	1	1
12	4	6	1	1
13	4	7	1	1
14	4	8	1	1
15	1	1	1	2
16	1	2	1	2
17	2	1	1	2
18	2	2	1	2
19	2	3	1	2
20	2	5	1	2
21	3	2	1	2
22	3	4	1	2
23	3	5	1	2
24	3	6	1	2
25	4	5	1	2
26	4	6	1	2

Fig. 3: source-detector layout

C. *Get Source-Detector Distance and Visualize Optodes*

```
Channel 1 distance: 30.02 mm
Channel 2 distance: 30.00 mm
Channel 3 distance: 30.00 mm
Channel 4 distance: 30.02 mm
```

Fig. 4: Source-detector distance.

Visualize the optodes and the layout of the fNIRS channels in 3D. This will help you understand the relative positioning of the optodes in the setup. Use visualization tools in MATLAB to achieve this.

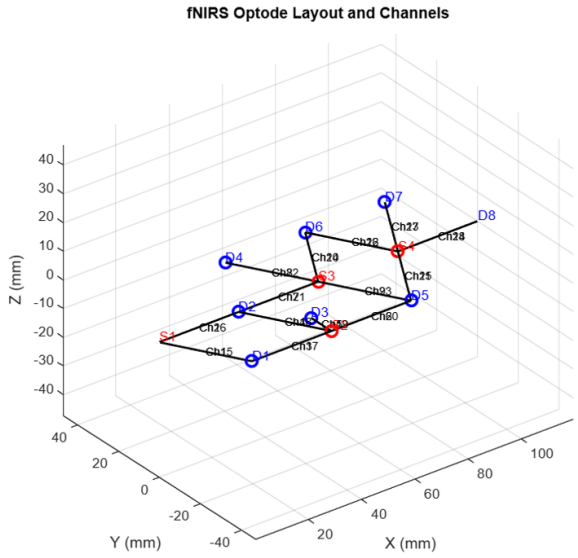

Fig. 5: Optodes placement visualization.

V. Preprocessing Steps

Note: Familiarity with digital signal processing (DSP) is recommended for understanding the preprocessing steps.

A. *Visual Inspection*

Visual inspection is crucial for identifying noisy channels or time windows to exclude from analysis.

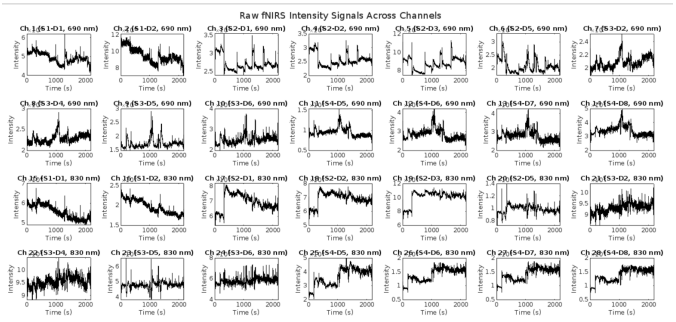

Fig. 6: subject 5

1) Visual Inspection on subject level: : we can discard the whole subject as the signal is corrupted.

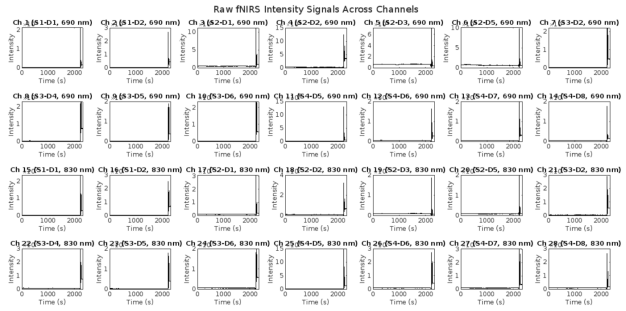

Fig. 7: Visual inspection on subject level ”sub30”.

2) Visual Inspection on channel level:: we can discard channel due to high noise as ch2

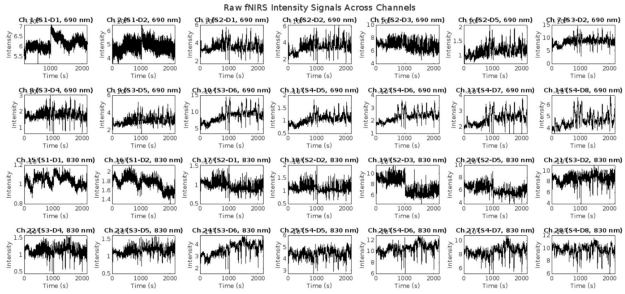

Fig. 8: Visual inspection on channel level”sub1”.

B. Convert to Optical Density

The raw intensity data is converted to optical density using the Beer–Lambert law. This step helps to linearize the raw data and prepare it for further processing.

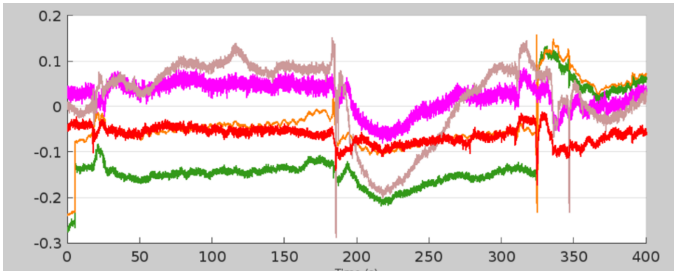

Fig. 9: Conversion of raw light intensity to optical density.

C. Motion Correction

Motion artifacts are common in fNIRS data and can significantly affect signal quality. Various techniques are available to address these artifacts:

1) Common Techniques for Motion Artifact Correction:

- **Wavelet Filtering**: Suppresses motion artifacts by removing outlier coefficients in the wavelet domain.
- **Principal Component Analysis (PCA)**: Identifies and removes components associated with motion.
- **Spline Interpolation**: Interpolates motion-contaminated segments of the signal using smooth spline curves.
- **Kalman Filtering**: Applies a recursive filter to estimate the true signal while minimizing the impact of motion-induced noise.

2) Spline-Based Motion Correction:

- Uses spline interpolation to correct segments of the signal contaminated by motion.
- Produces a corrected dataset by smoothly interpolating the affected segments and preserving the underlying signal.

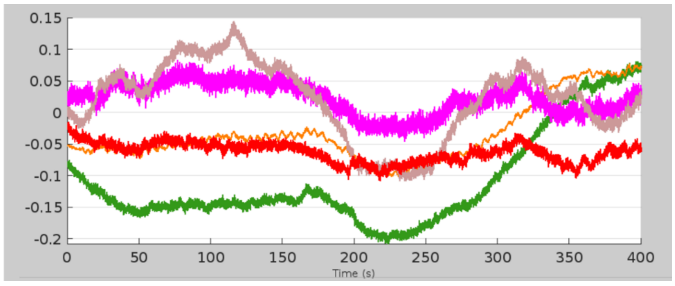

Fig. 10: Spline-Based Motion Correction.

3) Wavelet-Based Motion Correction: Utilizes a statistical threshold to detect and eliminate motion artifacts in the wavelet-transformed data.

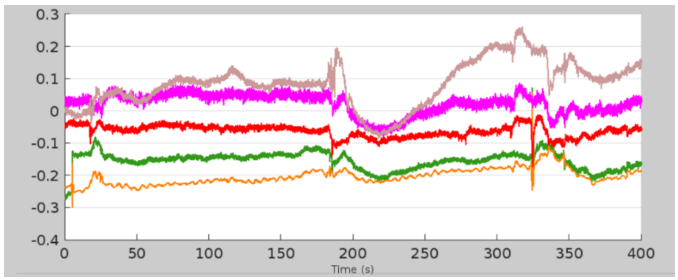

Fig. 11: Wavelet-based motion correction method.

D. Filtering Techniques

Use frequency-based filters (low pass, band pass, high pass) to eliminate unwanted noise.

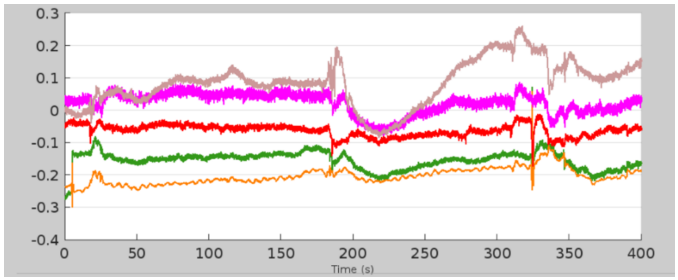

Fig. 12: Frequency-based filters for signal cleaning.

PCA filters can also be applied to remove high variance components.

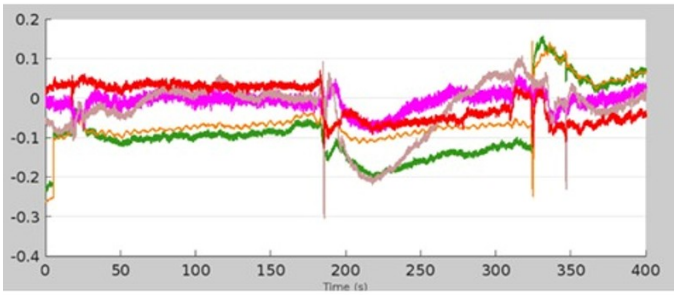

Fig. 13: PCA filters for signal cleaning.

E. Baseline Correction

Methods like bandpass filtering and baseline subtraction help to remove low-frequency drifts from the data.

F. Modified Beer-Lambert Law (MBLL)

The optical density is converted to concentration changes of:

- HbO (oxyhaemoglobin)

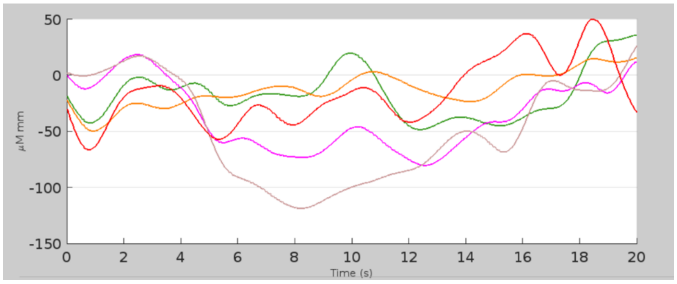

Fig. 14: HbO

- HbR (deoxyhaemoglobin)

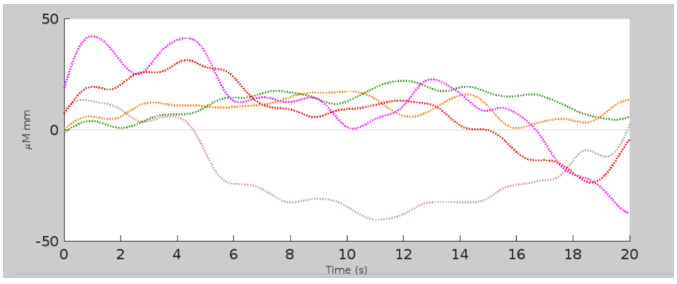

Fig. 15: HbR

- HbT (total haemoglobin)

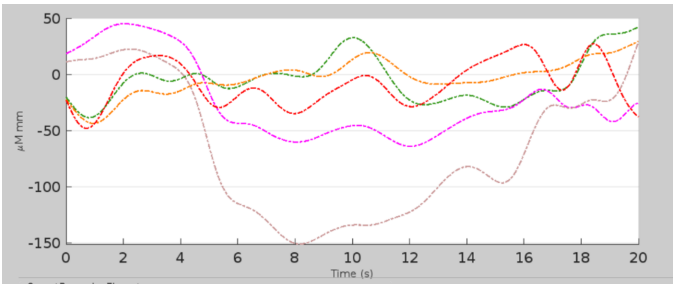

Fig. 16: HbT

G. Short Separation Channel

Use short-separation channels for GLM to regress out the component of the light source (LS) signal correlated with the short-separation (SS) signal.

VI. Statistical Analysis

A. Core Questions in fNIRS Analysis

Some core questions in fNIRS analysis include:

- Which brain areas were activated?
- Are there differences between conditions or groups?

B. Statistical Models

1) 1. Block Averaging: Block averaging is ideal for simple experimental designs and allows for a visual inspection of hemodynamic responses before applying more complex models.

2) 2. Generalized Linear Models (GLM): GLMs are robust statistical models used to estimate task-related activations. The model is:

$$Y = X\beta + \epsilon$$

Where:

- Y is the observed data
- X is the design matrix (predictors)
- β is the parameters to estimate
- ϵ is the error term

3) 3. Linear Mixed Models (LMMs): LMMs extend GLMs to account for both fixed and random effects. These models are suitable for repeated-measures or hierarchical data, and they help account for individual variability.

VII. Conclusion

This pipeline provides a comprehensive approach to fNIRS data analysis, from raw data inspection to statistical modeling. By following these steps, researchers can ensure accurate analysis of hemodynamic responses and investigate brain activity reliably.

Acknowledgements

I would like to thank my supervisor, Dr. Meena Makary, for his guidance and support throughout the project.

APPENDIX

A. fNIRS Data Preprocessing and Motion Correction

```
% Load subject data
sub_1 = matfile('S1001_run01.nirs');
whos(sub_1);

SD = sub_1.SD;
wavelengths = SD.Lambda;

ml = sub_1.ml;

% Get source-detector distance
SD = sub_1.SD;
wavelengths = SD.Lambda;
ml = sub_1.ml;
srcPos = SD.SrcPos;
detPos = SD.DetPos;
for ch = 1:4
    s_idx = ml(ch, 1);
    d_idx = ml(ch, 2);
    dist = norm(srcPos(s_idx,:) - detPos(d_idx, :));
         % Euclidean distance
    fprintf('Channel %d distance: %.2f mm\n', ch,
        dist);
end

% Visualizing the optodes
srcPos = SD.SrcPos;
detPos = SD.DetPos;

figure;
hold on;
axis equal;
title('fNIRS Optode Layout and Channels');
xlabel('X (mm)');
ylabel('Y (mm)');
zlabel('Z (mm)');

% Plot sources in red
for i = 1:size(srcPos, 1)
    plot3(srcPos(i,1), srcPos(i,2), srcPos(i,3), 'ro
        ', 'MarkerSize', 8, 'LineWidth', 2);
    text(srcPos(i,1), srcPos(i,2), srcPos(i,3)+2,
        sprintf('S%d', i), 'Color', 'r');
end

% Plot detectors in blue
for i = 1:size(detPos, 1)
    plot3(detPos(i,1), detPos(i,2), detPos(i,3), 'bo
        ', 'MarkerSize', 8, 'LineWidth', 2);
    text(detPos(i,1), detPos(i,2), detPos(i,3)+2,
        sprintf('D%d', i), 'Color', 'b');
end

% Plot channels (source-detector pairs) as lines
for ch = 1:size(ml, 1)
    s_idx = ml(ch, 1);
    d_idx = ml(ch, 2);

    % Coordinates
    s_pos = srcPos(s_idx, :);
    d_pos = detPos(d_idx, :);

    % Plot line
    plot3([s_pos(1), d_pos(1)], [s_pos(2), d_pos(2)
        ], [s_pos(3), d_pos(3)], 'k-', 'LineWidth',
        1.5);

    % Channel midpoint label
    mid = (s_pos + d_pos) / 2;
    text(mid(1), mid(2), mid(3), sprintf('Ch%d', ch)
        , 'Color', 'k', 'FontSize', 8);
end

grid on;
view(3);  % 3D view

% Visualize the Data
d = sub_1.d;
t = sub_1.t;
ml = sub_1.ml;
wavelengths = [690, 830];  % Known wavelengths

nChannels = size(d, 2);

figure;
set(gcf, 'Name', 'Raw fNIRS Data - Visual Inspection
    ', 'NumberTitle', 'off');
tiledlayout('flow');

for ch = 1:nChannels
    nexttile;
    plot(t, d(:, ch), 'k');
    xlabel('Time (s)');
    ylabel('Intensity');

    s = ml(ch,1);
    d_idx = ml(ch,2);
    lambda_idx = ml(ch,4);  % Adjust if needed
    lambda = wavelengths(lambda_idx);

    title(sprintf('Ch %d (S%d-D%d, %d nm)', ch, s,
        d_idx, lambda));
end

sgtitle('Raw fNIRS Intensity Signals Across Channels
    ');

% Plot specific channels
figure;
plot(t, d(:,1:5));   % show first 4 channels
xlabel('Time (s)');
ylabel('Intensity');
title('Raw Light Intensity - Channels 1 to 5');
legend('Ch 1', 'Ch 2', 'Ch 3', 'Ch 4', 'Ch 5');

% Convert raw intensity to optical density
dod = hmrIntensity2OD(d);  % Homer2 function

% Plot OD for first 5 channels
figure;
plot(t, dod(:,1:5));
xlabel('Time (s)');
ylabel('Optical Density (a.u.)');
title('Optical Density - Channels 1 to 5');
legend('Ch 1', 'Ch 2', 'Ch 3', 'Ch 4', 'Ch 5');

% Wavelet-based motion correction
iqr_thresh = 1.5; % Interquartile range threshold (
    suggested: 1.5  3  )

% Run Wavelet-based correction
[d_wavelet, ~] = hmrMotionCorrectWavelet(data.d,
    data.t, iqr_thresh);

% Replace original data with corrected data
data.d = d_wavelet;

% Save result
save('S1030_run01_WaveletCorrected.nirs', 'data');
```

Listing 1: fNIRS Data Preprocessing and Motion Correction

B. Stimulus Visualization

```
% Plot the AUX channels to inspect
aux = finfo.aux;

figure;
plot(t, aux(:,1), 'r'); hold on;
plot(t, aux(:,2), 'b');
xlabel('Time (s)');
legend('aux1 - Stimulus onset', 'aux2 - Response cue
    ');
title('AUX channels inspection');

% Detect stimulus onsets in AUX1
threshold = 0.5 * max(aux(:,1));  % Half the max
    value is safe
stim_onsets = find(aux(:,1) > threshold);

% Get only the rising edges
stim_onsets = stim_onsets([true; diff(stim_onsets) >
     10]); % 10 samples apart minimum

% Convert to time
stim_times = t(stim_onsets);

fprintf('Detected %d stimulus onsets.\n', numel(
    stim_times));

% Create a new stimulus matrix "s"
s = zeros(length(t),1);  % initialize

% Insert triggers at stimulus onset times
for i = 1:length(stim_onsets)
    s(stim_onsets(i)) = 1;   % mark the event
end

% Visualize the reconstructed stimulus
figure;
plot(t, aux(:,1)*0.5, 'r'); hold on;
plot(t, s, 'k');
legend('aux1 - Original', 's - Reconstructed
    stimulus');
xlabel('Time (s)');
title('Original AUX and Reconstructed Stimulus (s)')
    ;
```

Listing 2: Stimulus Detection and Visualization