# NeuroDOT 2.2.0 Base User Manual



# Table of Contents

[NeuroDOT 2.2.0 Base User Manual 1](#_Toc514411138)

[Table of Contents 2](#_Toc514411139)

[Introduction 3](#_Toc514411140)

[Purpose 3](#_Toc514411141)

[What Is Base? 3](#_Toc514411142)

[Feedback 3](#_Toc514411143)

[Getting Started 4](#_Toc514411144)

[Installation 4](#_Toc514411145)

[What Is In Base? 4](#_Toc514411146)

[Data 4](#_Toc514411147)

[Documentation 4](#_Toc514411148)

[Functions 4](#_Toc514411149)

[Support\_Files 4](#_Toc514411150)

[Learn Base With The Tutorial 4](#_Toc514411151)

[What Next? 5](#_Toc514411152)

[Toolbox Reference 6](#_Toc514411153)

[Index 6](#_Toc514411154)

# Introduction

## Purpose

This manual is intended to introduce users to NeuroDOT 2.2.0 Base (or just “Base” for short) and serve as a reference for its use and documentation.

## What Is Base?

NeuroDOT 2.2.0 Base contains the full preprocessing and imaging pipelines, some of the light modeling pipeline, a basic selection of post-processing pipelines, 2D surface activation visualizations, and file support for NIFTI, HOMER, AFNI. It is the first full public release, with versioning and source control to standardize between editions. Base finalizes the core design standards, incorporates internal feedback, and has been prepared for public release. This toolbox can be used to preprocess, reconstruct, and postprocess voxel-space time series for any NIRS data, and contains the support files, samples, and documentation needed to create similar profiles for any device.

## Feedback

We are primarily looking for feedback on (1) errors and bugs in functions, and (2) confusing, incomplete, or difficult-to-use documentation, which includes this manual. However, all feedback is welcome!

Please give us any feedback you have via email, to David Muccigrosso ([muccigrosso.david@wustl.edu](mailto:muccigrosso.david@wustl.edu)), and CC Adam Eggebrecht at ([aeggebre@wustl.edu](mailto:aeggebre@wustl.edu)). Make sure to include “NeuroDOT 2.2.0 Base” in the subject line for referencing and archiving purposes, which helps us ensure that your concerns are addressed in a timely manner.

# Getting Started

## Installation

*Note: NeuroDOT is optimized for use in MATLAB 2015b on Windows and Unix. If you do not have MATLAB, contact your IT department or go to www.mathworks.com for more information on MATLAB. Your results may vary with other versions or operating systems.*

To install NeuroDOT 2.2.0 Base, copy the unzipped folder into the directory of your choosing. Then open MATLAB and either click on the "Set Path" button on the “Environment” section of the "Home" tab of the main MATLAB console, or copy and paste the following command into your command line, substituting the red bracketed words for the path into which you unzipped NeuroDOT:

**installpath = '[your chosen directory]';  
addpath(genpath(installpath))**

## What Is In Base?

The main Base folder contains a readme.txt file and four subfolders: Data, Documentation, Functions, and Support\_Files. All of the files contained in these folders are described in further detail in the Index portion of this manual, but here is a brief summary.

### Data

Data contains nine sample data sets in it, corresponding to neurology experiment paradigms that will be explained in the main tutorial: three for the hearing words paradigm, abbreviated as “HW”, one retinotopy sample for the A and C quadrants, abbreviated “AC”, one reading words sample, abbreviated as “RW”, and a single session of four retinotopy samples, abbreviated “CCW”, “CW”, “IN”, and “OUT”. There is also a subfolder with the original raw scan files for each of these samples.

### Documentation

Documentation contains the manuals, tutorials, appendices, and MATLAB help browser files. These instruct the user how to work with Base and serve as a reference for its specific functionalities.

*Warning: If their filenames are edited, the MATLAB help browser files may conflict with the function files in MATLAB’s search path.*

### Functions

Functions contains the functions that make up Base. The specific scope of these functions in Base covers preprocessing, image reconstruction, file IO, and data quality visualization.

### Support\_Files

Support\_Files contains pre-assembled files for use in file IO, reconstruction, and general reference of cap metadata.

## Learn Base With The Tutorial

The first place to start is the PowerPoint “NeuroDOT 2.2.0 Base Tutorial – Overview”. This tutorial explores a Hearing Words experimental paradigm and explains the data structures, files, and functions of NeuroDOT 2.2.0 Base. Further tutorials on the DOT Processing pipeline and assorted appendices delve into how to use those functions to process and reconstruct optical imaging data into functional neuroimaging spectroscopy volumes.

### What Next?

Upon completing the tutorials, you may be interested in the appendices. These cover detailed diagrams of the preprocessing pipeline, pipeline design, and “gold standard” examples of preprocessing results for the samples not used in the main tutorial.

# Toolbox Reference

## Index

[Files 13](#_Toc514411156)

[/readme.txt 13](#_Toc514411157)

[/NeuroDOT 2.2.0 Base Quick Start Guide.pptx 14](#_Toc514411158)

[/Data 15](#_Toc514411159)

[/NeuroDOT\_Base\_AC\_Sample.mat 15](#_Toc514411160)

[/NeuroDOT\_Base\_CCW\_Sample.mat 16](#_Toc514411161)

[/NeuroDOT\_Base\_CW\_Sample.mat 17](#_Toc514411162)

[/NeuroDOT\_Base\_GV\_Sample.mat 18](#_Toc514411163)

[/NeuroDOT\_Base\_HW\_Sample\_1.mat 19](#_Toc514411164)

[/NeuroDOT\_Base\_HW\_Sample\_2.mat 20](#_Toc514411165)

[/NeuroDOT\_Base\_HW\_Sample\_Noisy.mat 21](#_Toc514411166)

[/NeuroDOT\_Base\_IN\_Sample.mat 22](#_Toc514411167)

[/NeuroDOT\_Base\_OUT\_Sample.mat 23](#_Toc514411168)

[/NeuroDOT\_Base\_RW\_Sample.mat 24](#_Toc514411169)

[/HOMER\_Sample 25](#_Toc514411170)

[/test\_homer2.nirs 25](#_Toc514411171)

[/ND1\_Sample 26](#_Toc514411172)

[/NeuroDOT\_Base\_Converter\_Sample.mat 26](#_Toc514411173)

[/Raw\_Samples 27](#_Toc514411174)

[/AC\_Sample/ 27](#_Toc514411175)

[/CCW\_Sample/ 28](#_Toc514411176)

[/CW\_Sample/ 29](#_Toc514411177)

[/GV\_Sample/ 30](#_Toc514411178)

[/HW\_Sample\_1/ 31](#_Toc514411179)

[/HW\_Sample\_2/ 32](#_Toc514411180)

[/HW\_Sample\_Noisy/ 33](#_Toc514411181)

[/IN\_Sample/ 34](#_Toc514411182)

[/OUT\_Sample/ 35](#_Toc514411183)

[/RW\_Sample/ 36](#_Toc514411184)

[/Documentation 37](#_Toc514411185)

[/Appendices 37](#_Toc514411186)

[/Sample\_Results 37](#_Toc514411187)

[/NeuroDOT 2.2.0 Base Appendix – AC Sample Results.pptx 37](#_Toc514411188)

[/NeuroDOT 2.2.0 Base Appendix – GV Sample Results.pptx 38](#_Toc514411189)

[/NeuroDOT 2.2.0 Base Appendix – HW Sample 2 Results.pptx 39](#_Toc514411190)

[/NeuroDOT 2.2.0 Base Appendix – HW Sample Noisy Results.pptx 40](#_Toc514411191)

[/NeuroDOT 2.2.0 Base Appendix – RW Sample Results.pptx 41](#_Toc514411192)

[/NeuroDOT 2.2.0 Base Appendix – Atlases.pptx 42](#_Toc514411193)

[/NeuroDOT 2.2.0 Base Appendix – File IO.pptx 43](#_Toc514411194)

[/NeuroDOT 2.2.0 Base Appendix – Pipeline Diagrams.pptx 44](#_Toc514411195)

[/NeuroDOT 2.2.0 Base Appendix – Preprocessing Pipeline Analysis.pptx 45](#_Toc514411196)

[/MATLAB\_Help\_Files/ 46](#_Toc514411197)

[/Scripts 47](#_Toc514411198)

[/NeuroDOT\_Base\_DOT\_Processing\_Script.m 47](#_Toc514411199)

[/NeuroDOT\_Base\_Preprocessing\_Pipeline\_Script.m 48](#_Toc514411200)

[/NeuroDOT\_Base\_Reconstruction\_Pipeline\_Script.m 49](#_Toc514411201)

[/Tutorials 50](#_Toc514411202)

[/NeuroDOT 2.2.0 Base Tutorial – Overview.pptx 50](#_Toc514411203)

[/NeuroDOT 2.2.0 Base Tutorial – DOT Processing Pipeline.pptx 51](#_Toc514411204)

[/NeuroDOT 2.2.0 Base User Manual.docx 52](#_Toc514411205)

[/Functions 53](#_Toc514411206)

[/Analysis 53](#_Toc514411207)

[/BlockAverage.m 53](#_Toc514411208)

[/FindGoodMeas.m 54](#_Toc514411209)

[/File\_IO 55](#_Toc514411210)

[/Check4MissingData.m 55](#_Toc514411211)

[/converter\_data.m 56](#_Toc514411212)

[/converter\_HOMER\_to\_ND2.m 57](#_Toc514411213)

[/converter\_info.m 58](#_Toc514411214)

[/converter\_ND1\_to\_ND2.m 59](#_Toc514411215)

[/converter\_ND2\_to\_HOMER.m 60](#_Toc514411216)

[/converter\_ND2\_to\_ND1.m 61](#_Toc514411217)

[/Crop2Synch.m 62](#_Toc514411218)

[/InterpretPulses.m 63](#_Toc514411219)

[/InterpretStimSynch.m 64](#_Toc514411220)

[/InterpretSynchBeeps.m 65](#_Toc514411221)

[/Load\_AcqDecode\_Data.m 66](#_Toc514411222)

[/Load\_HOMER.m 67](#_Toc514411223)

[/LoadMulti\_AcqDecode\_Data.m 68](#_Toc514411224)

[/LoadVolumetricData.m 69](#_Toc514411225)

[/Make\_NativeSpace\_4dfp.m 70](#_Toc514411226)

[/Read\_4dfp\_Header.m 71](#_Toc514411227)

[/Read\_AcqDecode\_Header.m 72](#_Toc514411228)

[/Read\_NIFTI\_Header.m 73](#_Toc514411229)

[/ReadAux.m 74](#_Toc514411230)

[/ReadInfoTxt.m 75](#_Toc514411231)

[/Save\_HOMER.m 76](#_Toc514411232)

[/SaveVolumetricData.m 77](#_Toc514411233)

[/Write\_4dfp\_Header.m 78](#_Toc514411234)

[/Write\_NIFTI.m 79](#_Toc514411235)

[/GUIs 80](#_Toc514411236)

[/Cap\_Fitter.fig, /Cap\_Fitter.m 80](#_Toc514411237)

[/operate\_optodes.m 81](#_Toc514411238)

[/springfit\_optodes.m 82](#_Toc514411239)

[/Reconstruction 83](#_Toc514411240)

[/reconstruct\_img.m 83](#_Toc514411241)

[/smooth\_Amat.m 84](#_Toc514411242)

[/spectroscopy\_img.m 85](#_Toc514411243)

[/Tikhonov\_invert\_Amat.m 86](#_Toc514411244)

[/Spatial\_Transforms 87](#_Toc514411245)

[/affine3d\_img.m 87](#_Toc514411246)

[/change\_space\_coords.m 88](#_Toc514411247)

[/Good\_Vox2vol.m 89](#_Toc514411248)

[/rotate\_cap.m 90](#_Toc514411249)

[/rotation\_matrix.m 91](#_Toc514411250)

[/scale\_cap.m 92](#_Toc514411251)

[/Support 93](#_Toc514411252)

[/CheckOrientation.m 93](#_Toc514411253)

[/istablevar.m 94](#_Toc514411254)

[/Temporal\_Transforms 95](#_Toc514411255)

[/detrend\_tts.m 95](#_Toc514411256)

[/fft\_tts.m 96](#_Toc514411257)

[/gethem.m 97](#_Toc514411258)

[/highpass.m 98](#_Toc514411259)

[/logmean.m 99](#_Toc514411260)

[/lowpass.m 100](#_Toc514411261)

[/normalize2range\_tts.m 101](#_Toc514411262)

[/regcorr.m 103](#_Toc514411263)

[/resample\_tts.m 104](#_Toc514411264)

[/Visualizations 105](#_Toc514411265)

[/adjust\_brain\_pos.m 105](#_Toc514411266)

[/applycmap.m 107](#_Toc514411267)

[/PlotCap.m 108](#_Toc514411268)

[/PlotCapData.m 109](#_Toc514411269)

[/PlotCapGoodMeas.m 110](#_Toc514411270)

[/PlotCapMeanLL.m 111](#_Toc514411271)

[/PlotFalloffData.m 112](#_Toc514411272)

[/PlotFalloffLL.m 113](#_Toc514411273)

[/PlotGray.m 114](#_Toc514411274)

[/PlotGrayData.m 115](#_Toc514411275)

[/PlotHistogramData.m 116](#_Toc514411276)

[/PlotHistogramSTD.m 117](#_Toc514411277)

[/PlotInterpSurfMesh.m 118](#_Toc514411278)

[/PlotLRMeshes.m 119](#_Toc514411279)

[/PlotMeshSurface.m 120](#_Toc514411280)

[/PlotPowerSpectrumAllMeas.m 121](#_Toc514411281)

[/PlotPowerSpectrumData.m 122](#_Toc514411282)

[/PlotPowerSpectrumMean.m 123](#_Toc514411283)

[/PlotSlices.m 124](#_Toc514411284)

[/PlotSlicesMov.m 126](#_Toc514411285)

[/PlotSlicesTimeTrace.m 127](#_Toc514411286)

[/PlotTimeTraceAllMeas.m 128](#_Toc514411287)

[/PlotTimeTraceData.m 129](#_Toc514411288)

[/PlotTimeTraceMean.m 130](#_Toc514411289)

[/vol2surf\_mesh.m 131](#_Toc514411290)

[/Support\_Files 132](#_Toc514411291)

[/Raw\_Support\_Files/ 132](#_Toc514411292)

[/Spectroscopy/E.mat 132](#_Toc514411293)

[/A\_Adult\_96x92.mat 133](#_Toc514411294)

[/Atlas\_MNI152nl\_T1\_on\_111.mat 134](#_Toc514411295)

[/Atlas\_MNI152nl\_T1\_on\_333.mat 135](#_Toc514411296)

[/Atlas\_Segmented\_MNI152nl\_T1\_on\_111.mat 136](#_Toc514411297)

[/Atlas\_Segmented\_MNI152nl\_T1\_on\_333.mat 137](#_Toc514411298)

[/Cap\_Fitter\_Mesh\_Sample.mat 138](#_Toc514411299)

[/LR\_Meshes\_MNI\_164k.mat 139](#_Toc514411300)

[/Pad\_Adult\_96x92.mat 140](#_Toc514411301)

[/Pad\_AdultV24x28.mat 141](#_Toc514411302)

[/Pad\_Baby32x34.mat 142](#_Toc514411303)

[Data Structures 143](#_Toc514411304)

[data 143](#_Toc514411305)

[info 144](#_Toc514411306)

[info.io 144](#_Toc514411307)

[info.system 145](#_Toc514411308)

[info.paradigm 146](#_Toc514411309)

[info.pairs 147](#_Toc514411310)

[info.optodes 148](#_Toc514411311)

[info.tissue 149](#_Toc514411312)

[info.MEAS 150](#_Toc514411313)

[info.misc 151](#_Toc514411314)

[flags 152](#_Toc514411315)

[flags.Nsys 152](#_Toc514411316)

[flags.crop\_level 153](#_Toc514411317)

[flags.bthresh 154](#_Toc514411318)

[flags.gbox 155](#_Toc514411319)

[flags.gsigma 156](#_Toc514411320)

[flags.lambda1 157](#_Toc514411321)

[flags.lambda2 158](#_Toc514411322)

[flags.omega\_hp 159](#_Toc514411323)

[flags.omega\_lp1 160](#_Toc514411324)

[flags.omega\_lp2 161](#_Toc514411325)

[flags.omega\_lp3 162](#_Toc514411326)

[flags.omega\_resample 163](#_Toc514411327)

[flags.resample\_tol 164](#_Toc514411328)

[params 165](#_Toc514411329)

[params.fig\_size 165](#_Toc514411330)

[params.fig\_handle 167](#_Toc514411331)

[params.dimension 168](#_Toc514411332)

[params.rlimits 169](#_Toc514411333)

[params.Nnns 170](#_Toc514411334)

[params.Nwls 171](#_Toc514411335)

[params.useGM 172](#_Toc514411336)

[params.xlimits 173](#_Toc514411337)

[params.xscale 174](#_Toc514411338)

[params.ylimits 175](#_Toc514411339)

[params.yscale 176](#_Toc514411340)

[params.climits 177](#_Toc514411341)

[params.mode 178](#_Toc514411342)

[params.Nregs 179](#_Toc514411343)

[params.cbmode 180](#_Toc514411344)

[params.cblabels 181](#_Toc514411345)

[params.cbticks 182](#_Toc514411346)

[params.bins 183](#_Toc514411347)

[params.CH 184](#_Toc514411348)

[params.slices 185](#_Toc514411349)

[params.slices\_type 186](#_Toc514411350)

[params.orientation 187](#_Toc514411351)

[params.view 188](#_Toc514411352)

[params.ctx 189](#_Toc514411353)

[params.alpha 190](#_Toc514411354)

[params.OL 191](#_Toc514411355)

[params.DR 192](#_Toc514411356)

[params.Scale 193](#_Toc514411357)

[params.BG 194](#_Toc514411358)

[params.TC 195](#_Toc514411359)

[params.PD 196](#_Toc514411360)

[params.Cmap 197](#_Toc514411361)

[params.Th 198](#_Toc514411362)

[params.Saturation 199](#_Toc514411363)

[params.kernel 200](#_Toc514411364)

[A 201](#_Toc514411365)

[E 202](#_Toc514411366)

[dim 203](#_Toc514411367)

[dim.xmin, dim.xmax, dim.ymin, dim.ymax, dim.zmin, dim.zmax 203](#_Toc514411368)

[dim.xv, dim.yv, dim.zv 204](#_Toc514411369)

[dim.nVx, dim.nVy, dim.nVz 205](#_Toc514411370)

[dim.nVt 206](#_Toc514411371)

[dim.sV 207](#_Toc514411372)

[dim.mmppix 208](#_Toc514411373)

[dim.center 209](#_Toc514411374)

[dim.Good\_Vox 210](#_Toc514411375)

[header 211](#_Toc514411376)

[header.original\_header 211](#_Toc514411377)

[header.version\_of\_keys 212](#_Toc514411378)

[header.format 213](#_Toc514411379)

[header.conversion\_program 214](#_Toc514411380)

[header.filename 215](#_Toc514411381)

[header.bytes\_per\_pixel 216](#_Toc514411382)

[header.byte 217](#_Toc514411383)

[header.acq 218](#_Toc514411384)

[header.nDim 219](#_Toc514411385)

[header.nVx, header.nVy, header.nVz 220](#_Toc514411386)

[header.nVt 221](#_Toc514411387)

[header.mmx, header.mmy, header.mmz 222](#_Toc514411388)

[header.mmppix 223](#_Toc514411389)

[header.center 224](#_Toc514411390)

[mesh 225](#_Toc514411391)

[mesh.nodes 225](#_Toc514411392)

[mesh.elements 226](#_Toc514411393)

[mesh.data 227](#_Toc514411394)

[mesh.region 228](#_Toc514411395)

[ops 229](#_Toc514411396)

[ops.rot\_x, ops.rot\_y, ops.rot\_z 229](#_Toc514411397)

[ops.trans\_x, ops.trans\_y, ops.trans\_z 230](#_Toc514411398)

[ops.flip\_x, ops.flip\_y, ops.flip\_z 231](#_Toc514411399)

[ops.scale\_m 232](#_Toc514411400)

[Pipelines 233](#_Toc514411401)

[Preprocessing 233](#_Toc514411402)

[Reconstruction 235](#_Toc514411403)

Files

* /readme.txt
* /NeuroDOT 2.2.0 Base Quick Start Guide.pptx
* /Data
* /Documentation
* /Functions
* /Support\_Files

/readme.txt

**Description:**

A traditional “readme.txt” file in the time-honored format. Directs new users to this manual and advanced users to the Appendices and other resources.

/NeuroDOT 2.2.0 Base Quick Start Guide.pptx

**Description:**

A short (10-slide) guide to get new users up and running with the toolbox.

/Data

* /NeuroDOT\_Base\_AC\_Sample.mat
* /NeuroDOT\_Base\_CCW\_Sample.mat
* /NeuroDOT\_Base\_CW\_Sample.mat
* /NeuroDOT\_Base\_GV\_Sample.mat
* /NeuroDOT\_Base\_HW\_Sample\_1.mat
* /NeuroDOT\_Base\_HW\_Sample\_2.mat
* /NeuroDOT\_Base\_HW\_Sample\_Noisy.mat
* /NeuroDOT\_Base\_IN\_Sample.mat
* /NeuroDOT\_Base\_OUT\_Sample.mat
* /NeuroDOT\_Base\_RW\_Sample.mat
* /HOMER\_Sample
* /ND1\_Sample
* /Raw\_Samples

/NeuroDOT\_Base\_AC\_Sample.mat

**Description:**

Based on the data set from subject 1409, date 161110, acquisition AC002.

/NeuroDOT\_Base\_CCW\_Sample.mat

**Description:**

Based on the data set from subject 0813, date 101019, acquisition CCW.

/NeuroDOT\_Base\_CW\_Sample.mat

**Description:**

Based on the data set from subject 0813, date 101019, acquisition CW.

/NeuroDOT\_Base\_GV\_Sample.mat

**Description:**

Based on the data set from subject 1106, date 121023, acquisition GV.

/NeuroDOT\_Base\_HW\_Sample\_1.mat

**Description:**

Based on the data set from subject 1106, date 120702, acquisition HW.

/NeuroDOT\_Base\_HW\_Sample\_2.mat

**Description:**

Based on the data set from subject 1104, date 120618, acquisition HW1.

/NeuroDOT\_Base\_HW\_Sample\_Noisy.mat

**Description:**

Based on the data set from subject 1531, date 150810, acquisition HW001.

/NeuroDOT\_Base\_IN\_Sample.mat

**Description:**

Based on the data set from subject 0813, date 101019, acquisition IN.

/NeuroDOT\_Base\_OUT\_Sample.mat

**Description:**

Based on the data set from subject 0813, date 101019, acquisition OUT.

/NeuroDOT\_Base\_RW\_Sample.mat

Description:

Based on the data set from subject 1106, date 120702, acquisition RW.

/HOMER\_Sample

* /test\_homer2.nirs

/test\_homer2.nirs

**Description:**

This sample is provided for use in the File IO tutorial appendix. It was derived from the introductory sample used in the HOMER2 beginners’ guide.

/ND1\_Sample

* /NeuroDOT\_Base\_Converter\_Sample.mat

/NeuroDOT\_Base\_Converter\_Sample.mat

**Description:**

This sample is provided for use in the File IO tutorial appendix, based on HW Sample 1.

/Raw\_Samples

* /AC\_Sample/
* /CCW\_Sample/
* /CW\_Sample/
* /GV\_Sample/
* /HW\_Sample\_1/
* /HW\_Sample\_2/
* /HW\_Sample\_Noisy/
* /IN\_Sample/
* /OUT\_Sample/
* /RW\_Sample/

This folder contains the original raw scan files for the three NeuroDOT 2 samples. Each subfolder contains two subfolders – the system they were generated with creates two files for each scan, roughly corresponding to each half of the DOT cap – each of which contains (1) the main .mag raw data file, (2) the frame synch .fs file, (3) the –info.txt key metadata file, and (4) the .raw stimulus synchronization file.

/AC\_Sample/

**Description:**

This folder contains the raw files corresponding to subject 1409, date 161110, acquisition AC002.

/CCW\_Sample/

**Description:**

This folder contains the raw files corresponding to subject 0813, date 101019, acquisition CCW.

/CW\_Sample/

**Description:**

This folder contains the raw files corresponding to subject 0813, date 101019, acquisition CW.

/GV\_Sample/

**Description:**

This folder contains the raw files corresponding to subject 1106, date 121023, acquisition GV.

/HW\_Sample\_1/

**Description:**

This folder contains the raw files corresponding to subject 1106, date 120702, acquisition HW.

/HW\_Sample\_2/

**Description:**

This folder contains the raw files corresponding to subject 1104, date 120618, acquisition HW1.

/HW\_Sample\_Noisy/

**Description:**

This folder contains the raw files corresponding to subject 1531, date 150810, acquisition HW001.

/IN\_Sample/

**Description:**

This folder contains the raw files corresponding to subject 0813, date 101019, acquisition IN.

/OUT\_Sample/

**Description:**

This folder contains the raw files corresponding to subject 0813, date 101019, acquisition OUT.

/RW\_Sample/

**Description:**

This folder contains the raw files corresponding to subject 1106, date 120702, acquisition RW.

/Documentation

* /Appendices
* /MATLAB\_Help\_Files
* /Scripts
* /Tutorials
* /NeuroDOT 2.2.0 Base User Manual.docx

/Appendices

* /Sample\_Results
* /NeuroDOT 2.2.0 Base Appendix – Atlases.pptx
* /NeuroDOT 2.2.0 Base Appendix – File IO.pptx
* /NeuroDOT 2.2.0 Base Appendix – Pipeline Diagrams.pptx
* /NeuroDOT 2.2.0 Base Appendix – Preprocessing Pipeline Analysis.pptx

/Sample\_Results

* /NeuroDOT 2.2.0 Base Appendix – AC Sample Results.pptx
* /NeuroDOT 2.2.0 Base Appendix – GV Sample Results.pptx
* /NeuroDOT 2.2.0 Base Appendix – HW Sample 2 Results.pptx
* /NeuroDOT 2.2.0 Base Appendix – HW Sample Noisy Results.pptx
* /NeuroDOT 2.2.0 Base Appendix – RW Sample Results.pptx

/NeuroDOT 2.2.0 Base Appendix – AC Sample Results.pptx

**Description:**

The AC Results Appendix is intended for advanced users who would like to compare their own results to a set of results that have been verified by the NeuroDOT development team.

/NeuroDOT 2.2.0 Base Appendix – GV Sample Results.pptx

**Description:**

The GV Results Appendix is intended for advanced users who would like to compare their own results to a set of results that have been verified by the NeuroDOT development team.

/NeuroDOT 2.2.0 Base Appendix – HW Sample 2 Results.pptx

**Description:**

The HW Sample 2 Results Appendix is intended for advanced users who would like to compare their own results to a set of results that have been verified by the NeuroDOT development team.

/NeuroDOT 2.2.0 Base Appendix – HW Sample Noisy Results.pptx

**Description:**

The HW Sample Noisy Results Appendix is intended for advanced users who would like to compare their own results to a set of results that have been verified by the NeuroDOT development team.

/NeuroDOT 2.2.0 Base Appendix – RW Sample Results.pptx

**Description:**

The RW Sample Results Appendix is intended for advanced users who would like to compare their own results to a set of results that have been verified by the NeuroDOT development team.

/NeuroDOT 2.2.0 Base Appendix – Atlases.pptx

**Description:**

The Atlases Appendix is intended for users who may not be familiar with the topic. It lays out, in succinct and clear terms, a number of concepts surrounding anatomical atlases, how they are made, and related topics in spatial registration.

/NeuroDOT 2.2.0 Base Appendix – File IO.pptx

**Description:**

The File IO Appendix is intended for all users. It gives a brief overview of how to use the File IO functions in NeuroDOT 2.

/NeuroDOT 2.2.0 Base Appendix – Pipeline Diagrams.pptx

**Description:**

The Pipeline Diagrams Appendix is intended for advanced users who have completed the Preprocessing and Reconstruction Tutorial. It contains diagrams for the pipelines from a “high” level programming perspective and on a “functional” level.

/NeuroDOT 2.2.0 Base Appendix – Preprocessing Pipeline Analysis.pptx

**Description:**

The Preprocessing Pipeline Analysis Appendix is intended for advanced users who have completed the Preprocessing Tutorial. It goes over variations on the Preprocessing Pipeline to demonstrate the effect of removing various steps from the pipeline.

/MATLAB\_Help\_Files/

**Description:**

Contains MATLAB .m-files written for MATLAB Publisher to generate .html versions of in-function help sections for use in MATLAB’s Help Browser.

/Scripts

* /NeuroDOT\_Base\_DOT\_Processing\_Script.m
* /NeuroDOT\_Base\_Preprocessing\_Pipeline\_Script.m
* /NeuroDOT\_Base\_Reconstruction\_Pipeline\_Script.m

/NeuroDOT\_Base\_DOT\_Processing\_Script.m

**Description:**

The NeuroDOT Base DOT Processing Script is a template developed by the NeuroDOT team for basic preprocessing and reconstruction. A section of selected visualizations is left commented at the end of the script.

/NeuroDOT\_Base\_Preprocessing\_Pipeline\_Script.m

**Description:**

The NeuroDOT Base Preprocessing Pipeline Script is a template developed by the NeuroDOT team for the corresponding pipeline. A section of selected visualizations is left commented at the end of the script.

/NeuroDOT\_Base\_Reconstruction\_Pipeline\_Script.m

**Description:**

The NeuroDOT Base Reconstruction Pipeline Script is a template developed by the NeuroDOT team for the corresponding pipeline. A section of selected visualizations is left commented at the end of the script.

/Tutorials

* /NeuroDOT 2.2.0 Base Tutorial – Overview.pptx
* /NeuroDOT 2.2.0 Base Tutorial – DOT Preprocessing.pptx

/NeuroDOT 2.2.0 Base Tutorial – Overview.pptx

**Description:**

The Overview Tutorial is an introductory tutorial for users new to NeuroDOT. It explains the basics of near infrared spectroscopy (NIRS) and DOT techniques, and outlines the files, functions, and data structures of NeuroDOT 2.2.0 Base. Upon completion, the user will be familiarized with all of these components and ready to proceed to the other tutorials.

/NeuroDOT 2.2.0 Base Tutorial – DOT Processing Pipeline.pptx

**Description:**

The DOT Processing Tutorial is a tutorial for new users to NeuroDOT 2.2.0 Base. It assumes a basic knowledge of NeuroDOT. Upon completion, the user will be familiarized with the preprocessing and basic reconstruction of NIRS/DOT data.

/NeuroDOT 2.2.0 Base User Manual.docx

**Description:**

This document. See “Introduction” for more information.

/Functions

* /Analysis
* /File\_IO
* /Reconstruction
* /Spatial\_Transforms
* /Support
* /Temporal\_Transforms
* /Visualizations

/Analysis

* /BlockAverage.m
* /FindGoodMeas.m

/BlockAverage.m

**Description:**

This function takes a light-level array "data\_in" of the MEAS x TIME format, and uses the experiment information from "info" to cut that data timewise into blocks of equal length, which are then averaged together and output as "data\_out".

The “pulse” input allows the user to select which synchronization pulse to block average. The default is "info.paradigm.Pulse\_2", but other pulses can be designated with either a string ("Pulse\_3" or "4") or number (5) input.

The “dt” input allows the user to manually input the block length "dt", as either a string or number.

**Sample Code:**

data\_out = BlockAverage(data\_in, info, pulse, dt)

/FindGoodMeas.m

**Description:**

This function takes a light-level array "data" in the MEAS x TIME format, and calculates the variance of each channel as its noise level. These are then thresholded by the default value of 0.075 to create a logical array, and both the are returned as MEAS x 1 columns of the "info.MEAS" table. If pulse synch point information exists in “info.system.synchpts”, then FindGoodMeas will crop the data to the start and stop pulses.

The third input “bthresh” allows the user to specify a threshold value.

**Sample Code:**

info\_out = FindGoodMeas(data, info\_in, bthresh)

/File\_IO

* /Check4MissingData.m
* /converter\_data.m
* /converter\_HOMER\_to\_ND2.m
* /converter\_info.m
* /converter\_ND1\_to\_ND2.m
* /converter\_ND2\_to\_HOMER.m
* /converter\_ND2\_to\_ND1.m
* /Crop2Synch.m
* /InterpretPulses.m
* /InterpretStimSynch.m
* /InterpretSynchBeeps.m
* /Load\_AcqDecode\_Data.m
* /Load\_HOMER.m
* /LoadMulti\_AcqDecode\_Data.m
* /LoadVolumetricData.m
* /Make\_NativeSpace\_4dfp.m
* /Read\_4dfp\_Header.m
* /Read\_AcqDecode\_Header.m
* /Read\_NIFTI\_Header.m
* /ReadAux.m
* /ReadInfoTxt.m
* /Save\_HOMER.m
* /SaveVolumetricData.m
* /Write\_4dfp\_Header.m
* /Write\_NIFTI.m

/Check4MissingData.m

**Description:**

This function takes the -info.txt key file information and compares it to the .fs framesynch file. If any frames are missing samples, the user is asked to abort, exclude incomplete frame(s), or cut all samples after the first incomplete frame.

**Sample Code:**

[data\_out, info\_out, framepts\_out] = Check4MissingData(data\_in, info\_in, framepts\_in)

/converter\_data.m

**Description:**

This function reshapes any ND1 data array to ND2 format, or vice versa. The direction is determined by the string "conversion", and in the 'ND2 to ND1' direction, "info" is a required third input.

The "conversion" string input can be either 'ND2 to ND1' or 'ND1 to ND2'.

**Sample Code:**

data\_out = converter\_data(data\_in, conversion, info)

/converter\_HOMER\_to\_ND2.m

**Description:**

This function takes the "nirs" structure loaded from a HOMER ".nirs" file and converts its fields into the ND2 variables "data", "info", and optionally "aux".

**Sample Code:**

[data, info, aux] = converter\_HOMER\_to\_ND2(nirs)

/converter\_info.m

**Description:**

This function adapts the metadata structures between ND1 and ND2.

The "conversion" string input can be either 'ND2 to ND1' or 'ND1 to ND2'.

All leftover fields from ‘ND1 to ND2’ are stored in the "misc" field of the new structure.

Only intended for use with ND1 and ND2 data structures.

**Sample Code:**

info\_out = converter\_info(info\_in, conversion)

/converter\_ND1\_to\_ND2.m

**Description:**

This is a container function for converter\_data and converter\_info. This function is only intended for use with ND1 and ND2 data structures.

**Sample Code:**

[data\_out, info\_out] = converter\_ND1\_to\_ND2(data\_in, info\_in)

/converter\_ND2\_to\_HOMER.m

**Description:**

This function converts the ND2 variables "data" and "info" into HOMER2's ".nirs" format, and saves into a file specified by "filename" and path "pn".

**Sample Code:**

Save\_HOMER(data, info, filename, pn)

/converter\_ND2\_to\_ND1.m

**Description:**

This is a container function for converter\_data and converter\_info. This function is only intended for use with ND1 and ND2 data structures.

**Sample Code:**

[data\_out, info\_out] = converter\_ND2\_to\_ND1(data\_in, info\_in)

/Crop2Synch.m

**Description:**

This function crops a MEAS x TIME array "data\_in" to the pulses in "info.paradigm.synchpts", adjusts the synch points to the new scale, saves the original synch points, and returns both structures. This function uses the "flags" structure to specify loading parameters.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| crop\_level | (0 or 2) | Data cropping. 0 = none, 1 = start pulse, 2 = start and stop. |

**Sample Code:**

[data\_out, info\_out] = Crop2Synch(data\_in, info\_in, flags)

/InterpretPulses.m

**Description:**

This function reads the pulse frequencies in “info.paradigm.synchtype” and classifies them into pulses in fields “info.paradigm.Pulse\_1”, “info.paradigm.Pulse\_2”, etc.

**Sample Code:**

info.paradigm = InterpretPulses(info.paradigm);

/InterpretStimSynch.m

**Description**:

This function reads the frames "framepts" of an auxiliary synch file "raw\_synch", then calculates and returns synch block standard deviation and frequency information in "synch".

**Sample Code:**

synch = InterpretStimSynch(raw\_synch, framepts, info);

/InterpretSynchBeeps.m

**Description:**

This function uses the MATLAB function FINDPEAKS to convert the stim synch array "synch" into a set of pulse timepoints and pulse heights, which are returned in "synchpts" and "synchtype" respectively.

**Sample Code:**

[synchpts, synchtype] = InterpretSynchBeeps(synch);

/Load\_AcqDecode\_Data.m

**Description:**

This function reads a raw data file and all secondary files scan files in the directory specified by "filename". The raw data is validated and checked for missing frames, then reshaped into the ND2 format.

All secondary files must be located in the same folder as "filename".

The information is returned as a raw light level array "data", a metadata structure "info", stimulus synchronization information "synch", raw auxiliary file data "aux", and frame time point array "framepts".

Currently supports ".mag" and ".iq" files.

**Sample Code:**

[data, info, synch, aux, framepts] = Load\_AcqDecode\_Data (filename);

/Load\_HOMER.m

**Description:**

This function loads a HOMER2 file specified by "filename" and path "pn", and converts it into the ND2 format output as "data" and "info".

**Sample Code:**

[data, info] = Load\_HOMER(filename, pn)

/LoadMulti\_AcqDecode\_Data.m

**Description:**

This function finds any acquisitions "filename" located in directory "pn", and loads the raw data files and any other relevant scan files contained therein. The information is returned as a raw light level array "data", a metadata structure "info", stimulus synchronization information "synch", and raw auxiliary file data "aux".

It is assumed that all files use the AcqDecode format: "DATE-SUBJECT-TAGletter", with a trailing lowercase letter for multiple acquisitions. Also, separate acquisitions are stored in subfolders named by "DATEletter", EG "150115a" and "150115b".

Currently supports ".mag" and ".iq" files.

This function uses the "flags" structure to specify loading parameters.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| Nsys | 2 | Number of acquisitions. |

**Sample Code:**

[data, info, synch, aux] = LoadMulti\_AcqDecode\_Data (filename, pn, flags);

/LoadVolumetricData.m

**Description:**

This function loads a file specified by "filename", path "pn", and "file\_type", and returns it in two parts: the raw data file "volume", and the header "header", containing a number of key-value pairs in 4dfp format.

This function supports a full filename input, as long as the extension is included in the file name and matches a supported file type.

Supported File Types/Extensions: “.4dfp” 4dfp, “.nii” NIFTI.

Note: This function uses the NIFTI Reader toolbox available on MATLAB Central. This toolbox has been included with NeuroDOT 2.

**Sample Code:**

[volume, header] = LoadVolumetricData(filename, pn, file\_type);

/Make\_NativeSpace\_4dfp.m

**Description:**

This function checks whether the input header contains the "mmppix" and "center" fields (these are not always present in 4dfp files). If either is absent, a default called the "native space" is calculated from the other fields of the volume.  
  
**Sample Code:**

header\_out = Make\_NativeSpace\_4dfp(header\_in);

/Read\_4dfp\_Header.m

**Description:**

This function reads a .ifh text file specified by "filename", containing a number of key-value pairs. The specific pairs are parsed and stored as fields of the output structure "header".

**Sample Code:**

header = Read\_4dfp\_Header(filename);

/Read\_AcqDecode\_Header.m

**Description:**

This function reads the first bytes of an open binary NIRS file identified by "fid", and selects the correct format to load the rest of the header.

**Sample Code:**

[Nd, Ns, Nwl, Nt] = Read\_AcqDecode\_Header(fid);

/Read\_NIFTI\_Header.m

**Description:**

This function reads the structure "nii" and converts it to a 4dfp format header "header".

Note: The math in this function to calculate the "center" field for 4dfp is not straightforward, and is based on source code shared by Tim Coalson from his "nifti-4dfp" toolbox.

**Sample Code:**

header = Read\_NIFTI\_Header(nii);

/ReadAux.m

**Description:**

This function reads auxiliary channel files contained in the -info.txt key specified by "info.io.run", “filename”, and path “pn”.

The channel files must be located in the same directory as the raw data.

**Sample Code:**

aux = ReadAux(info, filename, pn);

/ReadInfoTxt.m

**Description:**

This function scans the -info.txt key file specified by "filename" and parses its contents into key-value pairs, which are returned in the "io" substructure of the main ND2 "info" structure. Most of the pairs are either only used in the loading process or kept for later troubleshooting if the data set shows inconsistencies.

**Sample Code:**

io = ReadInfoTxt(filename);

/Save\_HOMER.m

**Description**:

This function converts the ND2 variables "data" and "info" into HOMER2's ".nirs" format, and saves into a file specified by "filename" and path "pn".

**Sample Code:**

Save\_HOMER(data, info, filename, pn)

/SaveVolumetricData.m

**Description:**

This function saves volumetric data defined by "volume" and "header" into a file specified by "filename", path "pn", and "file\_type".

This function supports a full filename input, as long as the extension is included in the file name and matches a supported file type.

Supported File Types/Extensions: “.4dfp” 4dfp, “.nii” NIFTI.

Note: This function uses the NIFTI Reader toolbox available on MATLAB Central. This toolbox has been included with NeuroDOT 2.

**Sample Code:**

SaveVolumetricData(volume, header, filename, pn, file\_type)

/Write\_4dfp\_Header.m

**Description:**

This function writes the input "header" in 4dfp format to an .ifh file specified by "filename".

**Sample Code:**

Write\_4dfp\_Header(header, filename)

/Write\_NIFTI.m

**Description:**

This function converts the volumetric data "volume" described by "header" from 4dfp to NIFTI format, output as "nii".

Note: The math in this function to calculate the "center" field for 4dfp is not straightforward, and is based on source code shared by Tim Coalson from his "nifti-4dfp" toolbox.

**Sample Code:**

nii = Write\_NIFTI(volume, header, filename);

/GUIs

* /Cap\_Fitter.fig, /Cap\_Fitter.m
* /operate\_optodes.m
* /springfit\_optodes.m

/Cap\_Fitter.fig, /Cap\_Fitter.m

**Description:**

This is an interactive GUI that allows the user to load skull mesh and cap grid files, re-orient them with an affine transformation, and then run a spring-fit algorithm on them. The final result will then be saved in a file containing the original "info" structure, the resulting one, the mesh they were fit onto, and an "ops" structure recording the operations performed.

**Sample Code:**

Cap\_Fitter();

/operate\_optodes.m

**Description:**

This function takes the structure "info\_in" and performs the operations defined by the "ops" structure on the optode positions contained within. The results are output as "info\_out".

**Sample Code:**

info\_out = operate\_optodes(info\_in, ops)

/springfit\_optodes.m

**Description:**

This function applies an ORL-developed algorithm that fits a cap grid onto a mesh by minimizing a spring energy calculation.

**Sample Code:**

tpos = springfit\_optodes(mesh, rad, spos3, dpos3, anchorpts, anchor)

/Reconstruction

* /reconstruct\_img.m
* /smooth\_Amat.m
* /spectroscopy\_img.m
* /Tikhonov\_invert\_Amat.m

/reconstruct\_img.m

**Description:**

This function takes the inverted VOX x MEAS sensitivity matrix "iA" and right-multiplies it by the preprocessed MEAS x TIME light-level matrix "data" to reconstruct an image in voxel space. The image is output in a VOX x TIME matrix "img".

**Sample Code:**

img = reconstruct\_img(data, iA)

/smooth\_Amat.m

**Description:**

This function takes the inverted VOX x MEAS sensitivity matrix "iA\_in" and performs Gaussian smoothing in the 3D voxel space on each of the concatenated wavelength matrices within, returning it as “iA\_out”. The calculation is parallelized by MEAS, and only good voxels as defined by the "dim.Good\_Vox" field are used to reduce calculation time.

The last two optional inputs allow the user to specify the Gaussian filter size "gbox" and the filter width "gsigma". If no values or the empty matrix are supplied for these inputs, "gbox" and "gsigma" will default to values of 5 and 1.2, respectively.

The last input allows the user to smooth the sensitivity matrix in the MEAS x VOX orientation (for instance, if it is desired to perform smoothing before inversion) if "tflag" is 1.

**Sample Code:**

iA\_out = smooth\_Amat(iA\_in, dim, gbox, gsigma)

/spectroscopy\_img.m

**Description:**

This function takes the reconstructed VOX x TIME x WL light-level image "img\_in" and multiplies it by the inverse of the extinction coefficient matrix "E" to create an output VOX x TIME x HB matrix "img\_out", where HB 1 and 2 are the voxel-space time series images for HbR and HbO, respectively.

**Sample Code:**

img\_out = spectroscopy\_img(img\_in, E)

/Tikhonov\_invert\_Amat.m

**Description:**

This function takes a MEAS x VOX device sensitivity matrix "A" and performs a Tikhonov inversion, returning it as a VOX x MEAS matrix "iA".

The two optional inputs allow the user to specify the values of the "lambda1" and "lambda2" parameters in the inversion calculation. Their default values are 0.01 and 0.1, respectively.

**Sample Code:**

iA = Tikhonov\_invert\_Amat (A, info, lambda1, lambda2);

/Spatial\_Transforms

* /affine3d\_img.m
* /change\_space\_coords.m
* /Good\_Vox2vol.m
* /rotate\_cap.m
* /rotation\_matrix.m
* /scale\_cap.m

/affine3d\_img.m

**Description:**

This function takes a reconstructed, VOX x TIME image "imgA" and transforms it from its initial voxel space defined by the structure "infoA" into a target voxel space defined by the structure "infoB" and using the transform matrix "affine". The output is a VOX x TIME matrix "imgB" in the target voxel space.

The “interp\_type” input allows the user to specify an interpolation method for the INTERP3 function that AFFINE3D\_IMG uses. Other methods that can be used (input as strings) are 'nearest', 'spline', and 'cubic'. The default value is 'linear'.

**Sample Code:**

imgB = affine3d\_img(imgA, infoA, infoB, affine, interp\_type);

/change\_space\_coords.m

**Description:**

This function takes a set of coordinates "coord\_in" of the initial space "input\_type", and converts them into the new space defined by the structure "space\_info", which is then output as "coord\_out".

**Sample Code:**

coord\_out = change\_space\_coords(coord\_in, space\_info, input\_type)

/Good\_Vox2vol.m

**Description:**

This function reshapes a VOX x TIME array "img" into an X x Y x Z x TIME array "imgvol", according to the dimensions of the space described by "dim".

**Sample Code:**

imgvol = Good\_Vox2vol(img, dim)

/rotate\_cap.m

**Description:**

This function rotates the cap grid given by "tpos\_in" by the rotation vector "dTheta" (in degrees) and outputs it as "tpos\_out".

**Sample Code:**

tpos\_out = rotate\_cap(tpos\_in, dTheta)

/rotation\_matrix.m

**Description:**

This function generates a rotation matrix "rot" for the vector "direction" given an angle "theta" (in radians).

**Sample Code:**

rot = rotation\_matrix(direction, theta)

/scale\_cap.m

**Description:**

This function scales the full cap grid given in "tpos\_in" by a factor of "M" around its centroid, and outputs it as "tpos\_out".

**Sample Code:**

tpos\_out = scale\_cap(tpos\_in, M)

/Support

* /CheckOrientation.m
* /istablevar.m

/CheckOrientation.m

**Description:**

This function checks the directions of the data encoding in “info.optodes.plot3orientation” to make it compatible with 3D cap visualizations. The default reverses the x-axis and leaves the other two alone. The directions are returned in “xdir”, “ydir”, and “zdir”.

**Sample Code:**

[xdir, ydir, zdir] = CheckOrientation(info);

/istablevar.m

**Description:**

This function performs the same logical check on tables as ISFIELD does for structure arrays, by checking if the variable(s) "varname" exist within the table "table". "varname" must be either a string or a 1D cell array of strings. A horizontal cell array will give a logical array for each cell array element, while a vertical array will be equivalent to "ALL(ISTABLEVAR)".

Example: If info.pairs is a table with variable names 'Src', 'Det', 'WL', 'NN', and 'r', then,

ISTABLEVAR(info.pairs, {'WL'})

gives

ans = 1

and with a horizontal cell array,

ISTABLEVAR(info.pairs, {'WL', 'NN'})

gives

ans = 1 1

However, note that the component function STRCMP in ISTABLEVAR is case-sensitive, so,

ISTABLEVAR(info.pairs, {'wL', 'Nn'})

will give

ans = 0 0

and with a vertical input array will result in a single output, only returning "true" if both strings are matches.

ISTABLEVAR(info.pairs, {'WL'; 'NN'})

ans = 1

**Sample Code:**

if (isfield(info, 'MEAS') && ~istablevar(info.MEAS, 'GI'))

/Temporal\_Transforms

* /detrend\_tts.m
* /fft\_tts.m
* /gethem.m
* /highpass.m
* /logmean.m
* /lowpass.m
* /normalize2range\_tts.m
* /regcorr.m
* /resample\_tts.m

/detrend\_tts.m

**Description:**

This function takes a raw light-level data array "data\_in" of the format MEAS x TIME and removes the straight-line fit along the TIME dimension from each measurement, returning it as “data\_out”.

**Sample Code:**

data\_out = detrend\_tts(data\_in)

/fft\_tts.m

**Description:**

This function takes the input "data", pads it timewise to the next highest power of two (for better performance), performs the fast Fourier transform of each channel using the built-in MATLAB function FFT, normalizes by the padded time length, and takes the first half of the transformed data (which is the positive half of the frequency domain). The result is output as the magnitude of the Fourier transform.

There are several optional outputs as well.

1. The corresponding frequency domain, which extends from 0 to the Nyquist frequency, and is calculated from the input "framerate".
2. The power spectrum, which is the absolute value of the magnitude, squared.
3. The phase angle of the magnitude, as calculated by MATLAB’s ANGLE function.

**Sample Code:**

[ftmag, ftdomain, ftpower, ftphase] = fft\_tts(data, framerate)

/gethem.m

**Description:**

This function takes a light-level array "data" of the format MEAS x TIME, and using the scan metadata in "info.pairs" averages the shallow measurements for each wavelength present. The result is commonly referred to as the "hem" of a measurement set. If there is a good measurements logical vector present in "info.MEAS.GI", it will be applied to the data; otherwise, "info.MEAS.GI" will be set to true for all measurements (i.e., all measurements are assumed to be good). The variable "hem" is output in the format WL x TIME.

The last two inputs allow the user to set the criteria for determining shallow measurements. "sel\_type" can be 'r2d', 'r3d', or 'NN', corresponding to the columns of the "info.pairs" table, and "value" can either take the form of a two-element "[min, max]" vector (for 'r2d' and 'r3d'), or a scalar or vector containing all nearest neighbor numbers to be averaged. By default, this function averages the first nearest neighbor.

This hemodynamics data is the interfering signal component due to the blood vessels of the scalp. Since the first nearest neighbors are the shallowest measurements, averaging them gives a sense of the influence this signal has on the rest of the data.

**Sample Code:**

hem = gethem(data, info, sel\_type, value)

/highpass.m

**Description:**

This function takes a light-level array "data\_in" and applies to it a forward-backward, zero-phase digital highpass Butterworth filter at a Nyquist cutoff frequency of "omegaHz \* (2 \* frate)", where omegaHz is the normalized input cutoff frequency, and frate is the sampling rate, returning it as “data\_out”.

This function also removes the linear component of the input data.

**Sample Code:**

data\_out = highpass(data\_in, omegaHz, frate)

/logmean.m

**Description:**

This function takes a light-level data array "data\_in" and takes the negative log of each element of each row divided by that row's average. The result is output into “data\_out” in the same MEAS x TIME format.

The formal equation for the LOGMEAN operation is:

y = -ln(x / <x>)

The MATLAB pseudocode form is:

y(each row) = -log(x(each row) ./ mean(x(each row)));

Example: If data = [1, 10, 100; exp(1), 10\*exp(1), 100\*exp(1)];

then LOGMEAN(data) yields [3.6109, 1.3083, -.9943; 3.6109, 1.3083,

-.9943].

**Sample Code:**

data\_out = logmean(data\_in)

/lowpass.m

**Description:**

This function takes a light-level array "data\_in" and applies to it a forward-backward, zero-phase digital lowpass Butterworth filter at a Nyquist cutoff frequency of "omegaHz \* (2 \* frate)", where omegaHz is the normalized input cutoff frequency, and frate is the sampling rate, returning it as “data\_out”..

This function also removes the linear component of the input data.

**Sample Code:**

data\_out = lowpass(data\_in, omegaHz, frate)

/normalize2range\_tts.m

**Description:**

This function takes any input "data\_in" and calculates the mean of each channel for the time points specified by string "range\_to\_avg". If no range is specified, the function defaults to averaging ALL time points. Each channel then has this mean subtracted from it, and the result is output as "data\_out".

The formal equation for the normalize2range\_tts operation is:

y = x - <x(range)>

The MATLAB pseudocode form is:

y(each row) = x(each row) - mean(x(range));

Example 1: If data = [21, 20, 19; 300, 295, 305];

then normalize2range\_tts(data) yields [1, 0, -1; 0, -5, 5].

Example 2: If data = [1, 2, 3, 4, 5, 6, 7];

then normalize2range\_tts(data, '1:5') yields [-2, -1, 0, 1, 2, 3, 4].

**Sample Code:**

data\_out = normalize2range\_tts(data\_in, range\_to\_avg)

/regcorr.m

**Description:**

This function takes a light-level data array "data\_in", and using the scan metadata in "info.pairs" and a WL x MEAS "hem" hemodynamics array generated by GETHEM, performs a regression correction for each wavelength of the data, which is returned in the MEAS x TIME array "data\_out". The corresponding correlation coefficients for each measurement are returned in "R" as a MEAS x 1 array.

The formal equation for the regression is:

x\_hat = x - h \* hp \* x;

where x is the transpose (TIME x MEAS) of the original data for a single wavelength, h is the hem transpose (TIME x 1) for a single wavelength, hp is the pseudoinverse (1 x TIME) of h, and x\_hat is the estimation of the true x.

**Sample Code:**

[data\_out, R] = regcorr(data\_in, info, hem)

/resample\_tts.m

**Description:**

This function takes a raw light-level data array "data\_in" of the format MEAS x TIME, and resamples it (typically downward) to a new frequency using the built-in MATLAB function resample. The new sampling frequency is calculated as the ratio of input "omega\_resample" divided by "framerate" (both scalars), to within the tolerance specified by "tol".

This function is needed because the linear signal components, which can be important in other NeuroDOT pipeline calculations, can be inadvertently removed by downsampling using resample alone.

Note: This function resamples synch points in addition to data. Be sure to take care that your data and synch points match after running this function! "info.paradigm.init\_synchpts" stores the original synch points if you need to restore them.

**Sample Code:**

[data\_out, info\_out] = resample\_tts(data\_in, info\_in, omega\_resample, tol, framerate)

/Visualizations

* /adjust\_brain\_pos.m
* /applycmap.m
* /PlotCap.m
* /PlotCapData.m
* /PlotCapGoodMeas.m
* /PlotCapMeanLL.m
* /PlotFalloffData.m
* /PlotFalloffLL.m
* /PlotGray.m
* /PlotGrayData.m
* /PlotHistogramData.m
* /PlotHistogramSTD.m
* /PlotLRMeshes.m
* /PlotMeshSurface.m
* /PlotPowerSpectrumAllMeas.m
* /PlotPowerSpectrumData.m
* /PlotPowerSpectrumMean.m
* /PlotSlices.m
* /PlotSlicesMov.m
* /PlotSlicesTimeTrace.m
* /PlotTimeTraceAllMeas.m
* /PlotTimeTraceData.m
* /PlotTimeTraceMean.m
* /PlotVolMesh.m
* /vol2surf\_mesh.m

/adjust\_brain\_pos.m

**Description:**

This function takes the left and right hemispheric meshes "meshL" and "meshR", respectively, and repositions them to the proper perspective for display.

This function allows the user to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| ctx | ‘std’ | Defines inflation of mesh. |
| orientation | ‘t’ | Select orientation of volume. ‘t’ for transverse, ‘s’ for sagittal. |
| view | ‘lat’ | Sets the view perspective. |

**Sample Code:**

[Lnodes, Rnodes] = adjust\_brain\_pos(meshL, meshR, params)

/applycmap.m

**Description:**

This function fuses the N-D image array "img\_in" with a default flat gray background and applies a number of other default settings to create a scaled and colormapped N-D x 3 RGB image array "img\_out".

The second input fuses the image array "img\_in" with the anatomical atlas volume input "anat" as the background.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| TC | 0 | Direct map integer data values to defined color map ("True Color"). |
| DR | 1000 | Dynamic range. |
| Scale | 0.9 | Maximum value, as a decimal percentage, to which image is scaled. |
| PD | 0 | Declares that input image is positive definite. |
| Cmap |  | Color maps. |
| .P | ‘jet’ | Colormap for positive data values. |
| .N | (none) | Colormap for negative data values. |
| .flipP | 0 | Logical, flips the positive colormap. |
| .flipN | 0 | Logical, flips the negative colormap. |
| Th |  | Thresholds. |
| .P | (25% max) | Value of min threshold to display positive data values. |
| .N | (-Th.P) | Value of max threshold to display negative data values. |
| BG | [0.5, 0.5, 0.5] | Background color, as an RGB triplet. |

**Sample Code:**

img\_out = applycmap(img\_in, anat, params)

/PlotCap.m

**Description:**

This function plots a basic diagram of the cap based on the cap metadata stored in "info.optodes". All optodes are numbered; sources are colored light red, and detectors light blue.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [20, 200, 1240, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |

**Sample Code:**

PlotCap(info, params)

/PlotCapData.m

**Description:**

This function plots the input RGB information in one of three modes:

'text' - Optode numbers are arranged in a cap grid and colored with the RGB input.

'patch' - Optodes are plotted as patches and colored with the RGB input.

'textpatch' - Optodes are plotted as patches and colored with the RGB input, with optode numbers overlain in white.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [20, 200, 1240, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| mode | ‘textpatch’ | Display mode. |

**Sample Code:**

PlotCapData(SrcRGB, DetRGB, info, params)

/PlotCapGoodMeas.m

**Description:**

This function plots a visualization of the Good Measurements determined by FINDGOODMEAS and arranges them based on the metadata in "info.optodes". Good channels are depicted as green lines, bad channels red lines; sources and detectors are given lettering in light blue and red.

The plot title provides tallies for all specified groupings. The next line of the title lists how many optodes for which only 33% of their measurements are good. These optodes are surrounded with white circles.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [20, 200, 1240, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs averaged and displayed. |
| mode | ‘good’ | Display mode. ‘good’ displays channels above noise threshold, ‘bad’ below. |

**Sample Code:**

PlotCapGoodMeas(info, params)

/PlotCapMeanLL.m

**Description:**

This function plots an intensity map of the mean light levels for specified measurement groupings of each optode on the cap and arranges them based on the metadata in "info.optodes".

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [20, 200, 1240, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs averaged and displayed. |
| useGM | 0 | Use Good Measurements. |
| Cmap.P | ‘hot’ | Default color mapping. |

**Sample Code:**

PlotCapMeanLL(data, info, params)

/PlotFalloffData.m

**Description:**

This function takes one input array "fall\_data" and plots it against another "separations" to create a falloff plot.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| xlimits | ‘auto’ | Limits of x-axis. |
| xscale | ‘linear’ | Scaling of x-axis. |
| ylimits | ‘auto’ | Limits of y-axis. |
| yscale | ‘log’ | Scaling of y-axis. |

**Sample Code:**

PlotFalloffData(fall\_data, separation, params)

/PlotFalloffLL.m

**Description:**

This function takes a light-level array "data" of the MEAS x TIME format, and generates a plot of each channel's temporal mean against its source-detector distance, in the specified groupings.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs averaged and displayed. |
| useGM | 0 | Use Good Measurements. |
| xlimits | [0, 60] | Limits of x-axis. |
| xscale | ‘linear’ | Scaling of x-axis. |
| ylimits | [1e-6, 1e1] | Limits of y-axis. |
| yscale | ‘log’ | Scaling of y-axis. |

**Sample Code:**

PlotFalloffLL(data, info, params)

/PlotGray.m

**Description:**

This function takes a light-level array "data" of the MEAS x TIME format, and generates a grayscale plot of all the measurements in the specified groupings.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs displayed. |
| useGM | 0 | Use Good Measurements |
| climits | [-0.03, 0.03] | Limits of color axis. |

**Sample Code:**

PlotGray(data, info, params)

/PlotGrayData.m

**Description:**

This function plots the input "data" as a scaled image.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| climits | [data min and max] | Limits of color axis. |

**Sample Code:**

PlotGrayData(data, params)

/PlotHistogramData.m

**Description:**

This function plots a histogram of the input data.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| xlimits | ‘auto’ | Limits of x-axis. |
| ylimits | ‘auto’ | Limits of y-axis. |
| bins | [0:0.5:100] | Number or array of histogram bins. |

**Sample Code:**

h = PlotHistogramData(hist\_data, params)

/PlotHistogramSTD.m

**Description:**

This function plots a histogram of the distribution of channel variances for specified groupings. A red bar will show the threshold value.

The third input “bthresh” allows the user to specify a threshold value. If no input is supplied, the default value of 0.075 will be used.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs averaged and displayed. |
| xlimits | [0, 50] | Limits of x-axis. |
| ylimits | [0, 200] | Limits of y-axis. |
| bins | [0:0.5:100] | Number or array of histogram bins. |

**Sample Code:**

PlotHistogramSTD(info, params, bthresh)

/PlotInterpSurfMesh.m

**Description:**

This function takes functional imaging data "volume" and interpolates it onto the left and right hemispheric surface meshes given in "meshL" and "meshR", respectively, using the spatial information in "dim". The result is a bilateral sagittal view of the activations overlain onto the surface of the brain represented by the meshes.

This function allows the user to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| Scale | (90% max) | Maximum value to which image is scaled. |
| Th |  | Thresholds. |
| .P | (25% max) | Value of min threshold to display positive data values. |
| .N | (-Th.P) | Value of max threshold to display negative data values. |

**Sample Code:**

PlotInterpSurfMesh(volume, meshL, meshR, dim, params)

/PlotLRMeshes.m

**Description:**

This function renders the data in a pair of left and right hemispheric meshes "meshL.data" and "meshR.data", respectively, and applies full color mapping to them. If no data is present for either mesh, a default gray mesh will be plotted.

This function allows the user to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 960, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| Scale | (90% max) | Maximum value to which image is scaled. |
| PD | 0 | Declares that input image is positive definite. |
| cbmode | 0 | Specifies whether to use custom colorbar axis labels. |
| clabels | ([min max]) | Colorbar axis labels. When cbmode==1, min defaults to 0 if PD==1, both default to +/- Scale if supplied. When cbmode==0, then cblabels dictates colorbar axis limits. |
| cbticks | (none) | When cbmode==1, specifies positions of tick marks on colorbar axis. |
| alpha | 1 | Transparency of mesh. |
| view | ‘lat’ | Sets the view perspective. |

**Sample Code:**

PlotLRMeshes(meshL, meshR, params)

/PlotMeshSurface.m

**Description:**

This function creates a 3D visualization of the surface mesh "mesh", described by the space "infoVol".

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. |
| Nregs | (unique(mesh.region)) or 1 | Number of regions to display. If 0 is given, all region data will be ignored and a matte gray surface plotted. |
| TC | 1 | Direct map integer data values to defined color map ("True Color"). |
| Cmap.P | ‘jet’ | Colormap for positive data values. |
| BG | [0.8, 0.8, 0.8] | Background color, as an RGB triplet. |
| orientation | ‘t’ | Select orientation of volume, ‘t’ for transverse, ‘s’ for sagittal. |

Note: This function calls “applycmap”, which has further options for using "params" to specify parameters for the fusion, scaling, and colormapping process.

**Sample Code:**

PlotMeshSurface(mesh, infoVol, params)

/PlotPowerSpectrumAllMeas.m

**Description:**

This function takes a light-level array "data" of the MEAS x TIME format, and creates a plot of the absolute value squared of the FFT. Each individual time trace is plotted for specified groupings.

The last input allows the user to specify the data framerate.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs averaged and displayed. |
| useGM | 1 | Use Good Measurements |
| xlimits | [1e-3, 5] | Limits of x-axis. |
| xscale | ‘log’ | Scaling of x-axis. |
| ylimits | [0, 0.01] | Limits of y-axis. |
| yscale | ‘linear’ | Scaling of y-axis. |

**Sample Code:**

PlotPowerSpectrumAllMeas(data, info, params, info.framerate)

/PlotPowerSpectrumData.m

**Description:**

This function takes a light-level array "data" of the MEAS x TIME format, and creates a plot of the absolute value squared of the FFT.

The last input allows the user to specify the data framerate.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| xlimits | ‘auto’ | Limits of x-axis. |
| xscale | ‘log’ | Scaling of x-axis. |
| ylimits | ‘auto’ | Limits of y-axis. |
| yscale | ‘linear’ | Scaling of y-axis. |

**Sample Code:**

PlotPowerSpectrumData(data, info, params, info.framerate)

/PlotPowerSpectrumMean.m

**Description:**

This function takes a light-level array "data" of the MEAS x TIME format, and creates a plot of the absolute value squared of its FFT. A mean of this is taken for specified groupings.

The last input allows the user to specify the data framerate.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs averaged and displayed. |
| useGM | 1 | Use Good Measurements. |
| xlimits | [1e-3, 5] | Limits of x-axis. |
| xscale | ‘log’ | Scaling of x-axis. |
| ylimits | [0, 3e-5] | Limits of y-axis. |
| yscale | ‘linear’ | Scaling of y-axis. |

**Sample Code:**

PlotPowerSpectrumMean(data, info, params, info.framerate)

/PlotSlices.m

**Description:**

This function takes a 3D voxel space image "underlay" and generates views along the three canonical axes.

In interactive mode, left-click on any point to move to those slices. To reset to the middle of the volume, right-click anywhere. To cancel interactive mode, press "Q", "Esc", or the middle mouse button.

This function uses the volumetric data in "infoVol" to display spatial coordinates of the slices in question.

This function can accept a second 3D voxel space image in “overlay”. When this is done, all color mapping is applied to the overlay image, and the underlay is rendered as a grayscale image times the RGB triplet in “params.BG”.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [20, 200, 1240, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure |
| CH | 1 | Turns crosshairs on (1) and off (0). |
| Scale | (90% max) | Maximum value to which image is scaled. |
| PD | 0 | Declares that input image is positive definite. |
| cbmode | 0 | Specifies whether to use custom colorbar axis labels. |
| clabels | ([min max]) | Colorbar axis labels. When cbmode==1, min defaults to 0 if PD==1, both default to +/- Scale if supplied. When cbmode==0, then cblabels dictates colorbar axis limits. |
| cbticks | (none) | When cbmode==1, specifies positions of tick marks on colorbar axis. |
| slices | (center frames) | Select which slices are displayed. If empty, activates interactive navigation. |
| slices\_type | ‘idx’ | Use MATLAB indexing (‘idx’) for slices, or spatial coordinates (‘coord’) as provided by infoVol. |
| orientation | ‘t’ | Select orientation of volume, ‘t’ for transverse, ‘s’ for sagittal. |

Note: This function calls “applycmap”, which has further options for using "params" to specify parameters for the fusion, scaling, and colormapping process.

**Sample Code:**

PlotSlices(atlas, infoA, params, volume)

/PlotSlicesMov.m

**Description:**

This function uses the same basic inputs as the function PLOTSLICES to create a video of a 4D volume, saved under the filename "fn".

The last input allows the user to input an "info" structure or a framerate for the video to be recorded with. The default framerate is 1 Hz.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [20, 200, 1240, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure |

Note: This function calls “applycmap”, which has further options for using "params" to specify parameters for the fusion, scaling, and colormapping process.

**Sample Code:**

PlotSlicesMov(underlay, infoVol, params, volume, overlay, infoFR)

/PlotSlicesTimeTrace.m

**Description:**

This uses the same basic inputs as the function PlotSlices to create an interactive 3-axis plot of a 4D volume, plus an axis for the time trace of the selected voxel.

The last input allows the user to input an "info" structure to provide information about the 4D volume's native framerate. The default framerate is 1 Hz.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [20, 200, 1240, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure |
| CH | 1 | Turns crosshairs on (1) and off (0). |
| Scale | (90% max) | Maximum value to which image is scaled. |
| PD | 0 | Declares that input image is positive definite. |
| cbmode | 0 | Specifies whether to use custom colorbar axis labels. |
| clabels | ([min max]) | Colorbar axis labels. When cbmode==1, min defaults to 0 if PD==1, both default to +/- Scale if supplied. When cbmode==0, then cblabels dictates colorbar axis limits. |
| cbticks | (none) | When cbmode==1, specifies positions of tick marks on colorbar axis. |
| slices | (center frames) | Select which slices are displayed. If empty, activates interactive navigation. |
| slices\_type | ‘idx’ | Use MATLAB indexing (‘idx’) for slices, or spatial coordinates (‘coord’) as provided by infoVol. |
| orientation | ‘t’ | Select orientation of volume, ‘t’ for transverse, ‘s’ for sagittal. |
| kernel | [1] | A sampling kernel for the time trace plot. Other options: ‘gaussian’ | ‘cube’ | ‘sphere’. |

Note: This function calls “applycmap”, which has further options for using "params" to specify parameters for the fusion, scaling, and colormapping process.

**Sample Code:**

PlotSlicesTimeTrace(underlay, infoVol, params, overlay, info)

/PlotTimeTraceAllMeas.m

**Description:**

This function takes a light-level array "data" of the MEAS x TIME format, and plots its individual time traces in specified groupings.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs averaged and displayed. |
| useGM | 0 | Use Good Measurements. |
| xlimits | [0, Nt+1] | Limits of x-axis. |
| xscale | ‘linear’ | Scaling of x-axis. |
| ylimits | [1e-5 1e2] | Limits of y-axis. |
| yscale | ‘log’ | Scaling of y-axis. |

**Sample Code:**

PlotTimeTraceAllMeas(data, info, params)

/PlotTimeTraceData.m

**Description:**

This function takes a light-level array "data" of the MEAS x TIME format, and plots its time traces.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| xlimits | ‘auto’ | Limits of x-axis. |
| xscale | ‘linear’ | Scaling of x-axis. |
| ylimits | ‘auto’ | Limits of y-axis. |
| yscale | ‘linear’ | Scaling of y-axis. |

**Sample Code:**

PlotTimeTraceData(data, params)

/PlotTimeTraceMean.m

**Description:**

This function takes a light-level array "data" of the MEAS x TIME format, and plots the mean of its time traces in specified groupings.

This function uses the “params” structure to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| fig\_size | [200, 200, 560, 420] | Default figure position vector. |
| fig\_handle | (none) | Specifies a figure to target. If empty, spawns a new figure. |
| dimension | ‘2D’ | Specifies either a 2D or 3D plot rendering. |
| rlimits | (all R2D) | Limits of pair radii displayed. |
| Nnns | (all NNs) | Number of NNs displayed. |
| Nwls | (all WLs) | Number of WLs averaged and displayed. |
| useGM | 0 | Use Good Measurements. |
| xlimits | [0, Nt+1] | Limits of x-axis. |
| xscale | ‘linear’ | Scaling of x-axis. |
| ylimits | [-0.1, 0.1] | Limits of y-axis. |
| yscale | ‘linear’ | Scaling of y-axis. |

**Sample Code:**

PlotTimeTraceMean(data, info, params)

/vol2surf\_mesh.m

**Description:**

This function takes the mesh "mesh\_in" and interpolates the values of the volumetric data "volume" at the mesh's surface, using the spatial information in "dim". These values are overwritten in a new mesh, which is output as "mesh\_out".

This function allows the user to specify parameters for plot creation.

**Visualization Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Default | Effect |
| OL | 0 | If "overlap" data is presented (OL==1), this sets the interpolation method to "nearest". Default is "linear". |

**Sample Code:**

mesh\_out = vol2surf\_mesh(mesh\_in, volume, dim, params)

/Support\_Files

* /Raw\_Support\_Files/
  + /Spectroscopy/E.mat
* /A\_Adult\_96x92.mat
* /Atlas\_MNI152nl\_T1\_on\_111.mat
* /Atlas\_MNI152nl\_T1\_on\_333.mat
* /Atlas\_Segmented\_MNI152nl\_T1\_on\_111.mat
* /Atlas\_Segmented\_MNI152nl\_T1\_on\_333.mat
* /LR\_Meshes\_MNI\_164k.mat
* /Pad\_Adult\_96x92.mat
* /Pad\_AdultV24x28.mat
* /Pad\_Baby32x34.mat

/Raw\_Support\_Files/

**Description:**

This folder contains the original NeuroDOT 1 files from which the “Pad” and “Recon” support files were created.

/Spectroscopy/E.mat

**Description:**

This file contains an “E” matrix of spectroscopy coefficients that relate the light intensity to HbO and HbR concentrations via the modified Beer-Lambert law.

/A\_Adult\_96x92.mat

**Description:**

This file contains several variables required for image reconstruction with the ORL’s adult scanning cap: an “A” matrix for cap sensitivity, a “dim” structure describing the DOT space, and an “infoA” structure describing the sensitivity modeling space.

/Atlas\_MNI152nl\_T1\_on\_111.mat

**Description:**

This file contains a version of the MNI 152 non-linear T1 111 atlas that has been transformed into the DOT space. It is saved as a variable “atlas”.

/Atlas\_MNI152nl\_T1\_on\_333.mat

**Description:**

This file contains a version of the MNI 152 non-linear T1 333 atlas that has been transformed into the DOT space. It is saved as a variable “atlas”.

/Atlas\_Segmented\_MNI152nl\_T1\_on\_111.mat

**Description:**

This file contains a version of the MNI 152 non-linear T1 111 atlas segmentation that has been transformed into the DOT space. It is saved as a variable “atlas”.

/Atlas\_Segmented\_MNI152nl\_T1\_on\_333.mat

**Description:**

This file contains a version of the MNI 152 non-linear T1 333 atlas segmentation that has been transformed into the DOT space. It is saved as a variable “atlas”.

/Cap\_Fitter\_Mesh\_Sample.mat

**Description:**

A full-head sample mesh provided for use with the Cap\_Fitter GUI.

/LR\_Meshes\_MNI\_164k.mat

**Description:**

Contains two meshes, "MNIl" and "MNIr", corresponding to the left and right halves of the segmented MNI 152 non-linear T1 111 atlas.

/Pad\_Adult\_96x92.mat

**Description:**

This file contains an “info” structure with the “optodes”, “pairs”, and “tissue” substructures corresponding to the ORL’s adult scanning cap, with 96 sources and 92 detectors.

/Pad\_AdultV24x28.mat

**Description:**

This file contains an “info” structure with the “optodes”, “pairs”, and “tissue” substructures corresponding to the ORL’s adult visual cortex cap, with 24 sources and 28 detectors.

/Pad\_Baby32x34.mat

**Description:**

This file contains an “info” structure with the “optodes”, “pairs”, and “tissue” substructures corresponding to the ORL’s infant scanning cap, with 32 sources and 34 detectors.

Data Structures

* data
* info
* params
* A
* E
* dim
* header
* mesh

data

**Description:**

This variable is a MEAS x TIME sized array of raw optical data. Each row represents one measurement or channel, and each column represents one sampling point. MEAS is the total number of channels, and TIME is the total number of sampling points acquired.

info

**Description:**

This variable is a structure array containing scan metadata in several sub-structure fields:

* info.io
* info.system
* info.paradigm
* info.pairs
* info.optodes
* info.tissue
* info.MEAS
* info.misc

info.io

**Description:**

This sub-structure contains all information that is loaded during file IO but not used in NeuroDOT 2 – mainly key-value pairs from the –info.txt files. When adult cap files are loaded, this structure is subdivided into two fields, “a” and “b”, corresponding to information from each half of the adult cap.

This information is typically most useful when troubleshooting corrupted acquisitions.

info.system

**Description:**

This sub-structure contains information about the system used to collect the imaging data from the cap:

* PadName – The name of the cap used with the system.
* framerate – The current sampling rate of the data collection.
* init\_framerate – The initial sampling rate of the data collection.

info.paradigm

**Description:**

This sub-structure contains information on the experimental design of the scan:

* synchpts – Time points corresponding to events in the experiment, such as session start and stop or block starts.
* init\_synchpts – A backup of the original synch points.
* synchtype – An array of frequency values corresponding to each entry in synchpts. These frequency values distinguish the synch pulses from each other.
* Pulse\_1 – Indices of session start and stop times within synchpts
* Pulse\_2 – Indices of block start times within synchpts
* Pulse\_3/Pulse\_4 – Indices of extra stimuli or rest pulses within synchpts.

info.pairs

**Description:**

This field is a MEAS x 8 table of information describing source-detector measurements. The columns are:

* Src – Source number.
* Det – Detector number.
* NN – Nearest neighbor topology.
* WL – Wavelength number.
* lambda – Wavelength in nm.
* Mod – Modulation scheme (string).
* r2d – S-D pair radius on a 2D grid.
* r3d – S-D pair radius in 3D wrapped cap space.

info.optodes

**Description:**

This sub-structure contains information on the geometry of the imaging cap:

* CapName – Name of the grid file used to populate rest of sub-structure.
* dpos2 – List of detector positions on a 2D grid.
* dpos3 – List of detector positions in 3D wrapped cap space.
* spos2 – List of source positions on a 2D grid.
* spos3 – List of source positions in 3D wrapped cap space.
* plot3orientation – Structure containing the orientation of the cap space volume in string codes. These codes include the six cardinal biological directions used in neuroimaging: Right, Left, Posterior, Anterior, Dorsal, and Ventral.
  + i – Direction of the first dimension of the volume. Default is ‘R2L’.
  + j – Direction of the second dimension of the volume. Default is ‘P2A’.
  + k – Direction of the third dimension of the volume. Default is ‘D2V’.

info.tissue

**Description:**

This sub-structure contains information about light modeling. Its single field is “affine”, an affine transform matrix relating the DOT space to the atlas space.

info.MEAS

**Description:**

This sub-structure is an “on-the-fly” table mirroring “info.pairs”, containing information relevant to the MEAS dimension of the data.

* STD – Standard deviation of each channel.
* GI – Logical list based on application of a noise threshold to “info.MEAS.STD”.

info.misc

**Description:**

This sub-structure contains all of the fields of NeuroDOT 1 “info” structures that are not used in NeuroDOT 2. It will typically only be present in files that are converted using “converter\_ND1\_to\_ND2”.

flags

**Description:**

This structure allows the user to define and store parameters for pipelines.

* flags.Nsys
* flags.crop\_level
* flags.bthresh
* flags.gbox
* flags.gsigma
* flags.lambda1
* flags.lambda2
* flags.omega\_hp
* flags.omega\_lp1
* flags.omega\_lp2
* flags.omega\_lp3
* flags.omega\_resample
* flags.resample\_tol

flags.Nsys

**Description:**

Number of systems used to acquire data.

flags.crop\_level

**Description:**

Determines data cropping to synch points.

Key: 0 = none, 1 = crop to start pulse, 2 = start and stop pulses.

Defaults to 0 for Nsys == 1, and 2 for Nsys >= 2.

flags.bthresh

**Description:**

Noise threshold for Good Measurements analysis.

flags.gbox

**Description:**

Sensitivity matrix Gaussian smoothing filter width.

flags.gsigma

**Description:**

Sensitivity matrix Gaussian smoothing filter height.

flags.lambda1

**Description:**

First Tikhonov regularization parameter for sensitivity matrix inversion.

flags.lambda2

**Description:**

Second Tikhonov regularization parameter for sensitivity matrix inversion.

flags.omega\_hp

**Description:**

High Pass Filter cutoff frequency.

flags.omega\_lp1

**Description:**

First Low Pass Filter cutoff frequency.

flags.omega\_lp2

**Description:**

Second Low Pass Filter cutoff frequency.

flags.omega\_lp3

**Description:**

Third Low Pass Filter cutoff frequency.

flags.omega\_resample

**Description:**

Resampling frequency.

flags.resample\_tol

**Description:**

Tolerance threshold for resampling ratio.

params

**Description:**

This structure allows the user to define parameters for plot creation.

* params.fig\_size
* params.fig\_handle
* params.dimension
* params.rlimits
* params.Nnns
* params.Nwls
* params.useGM
* params.xlimits
* params.xscale
* params.ylimits
* params.yscale
* params.climits
* params.mode
* params.Nregs
* params.cbmode
* params.cblabels
* params.cbticks
* params.bins
* params.CH
* params.slices
* params.slices\_type
* params.orientation
* params.view
* params.ctx
* params.alpha
* params.OL
* params.DR
* params.Scale
* params.BG
* params.TC
* params.PD
* params.Cmap
* params.Th
* params.Saturation
* params.kernel

params.fig\_size

**Description:**

This field determines the size of the figure generated. The value follows the standard MATLAB position vector format: [left edge, bottom edge, width, height]

**Used By:**

All “Plot” functions

params.fig\_handle

**Description:**

This field specifies a figure to target. If empty, spawns a new figure.

**Used By:**

All “Plot” functions

params.dimension

**Description:**

This field specifies either a 2D or 3D plot rendering. When params.rlimits is being used, it also specifies whether to use info.pairs.r2D or info.pairs.r3D.

**Used By:**

PlotCap, PlotCapData, PlotCapMeanLL, PlotCapGoodMeas, PlotFalloffLL, PlotGray, PlotHistogramSTD, PlotPowerSpectrumAllMeas, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceMean

params.rlimits

**Description:**

This field determines a range of pair radii to be displayed, using the correct info.pairs column depending on whether the visualization is 2D or 3D. The value *must come in the form* of “[*rmin rmax*]”.

**Used By:**

PlotCapMeanLL, PlotCapGoodMeas, PlotFalloffLL, PlotGray, PlotHistogramSTD, PlotPowerSpectrumAllMeas, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceMean

params.Nnns

**Description:**

This field determines which nearest neighbor pairs are displayed. The value can be either a single number (“2”), or an array (“1:4”).

**Used By:**

PlotCapMeanLL, PlotCapGoodMeas, PlotFalloffLL, PlotGray, PlotHistogramSTD, PlotPowerSpectrumAllMeas, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceMean

params.Nwls

**Description:**

This field determines which wavelengths are displayed. The value must match those in the info.pairs.WL column, and may either be a single number (“1”), or an array (“1:2”).

**Used By:**

PlotCapMeanLL, PlotCapGoodMeas, PlotFalloffLL, PlotGray, PlotHistogramSTD, PlotPowerSpectrumAllMeas, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceMean

params.useGM

**Description:**

This field specifies whether to apply Good Measurements information to a visualization.

**Used By:**

PlotCapMeanLL, PlotFalloffLL, PlotGray, PlotPowerSpectrumAllMeas, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceMean

params.xlimits

**Description:**

This field determines the x-axis limits of most functions that plot data on an axis. The values must come in the form of “[*xmin xmax*]”.

**Used By:**

PlotFalloffData, PlotFalloffLL, PlotHistogramData PlotHistogramSTD, PlotPowerSpectrumAllMeas, PlotPowerSpectrumData, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceData, PlotTimeTraceMean

params.xscale

**Description:**

This field determines the scaling of the x-axis. Possible values are “linear” or “log”.

**Used By:**

PlotFalloffData, PlotFalloffLL, PlotPowerSpectrumAllMeas, PlotPowerSpectrumData, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceData, PlotTimeTraceMean

params.ylimits

**Description:**

This field determines the y-axis limits of most functions that plot data on an axis. The values must come in the form of “[*ymin ymax*]”.

**Used By:**

PlotFalloffData, PlotFalloffLL, PlotHistogramData PlotHistogramSTD, PlotPowerSpectrumAllMeas, PlotPowerSpectrumData, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceData, PlotTimeTraceMean

params.yscale

**Description:**

This field determines the scaling of the x-axis. Possible values are “linear” or “log”.

**Used By:**

PlotFalloffData, PlotFalloffLL, PlotPowerSpectrumAllMeas, PlotPowerSpectrumData, PlotPowerSpectrumMean, PlotTimeTraceAllMeas, PlotTimeTraceData, PlotTimeTraceMean

params.climits

**Description:**

This field determines the color axis limits of grayscale plots. The values must come in the form of “[c*min cmax*]”.

**Used By:**

PlotGray, PlotGrayData

params.mode

**Description:**

This field specifies the display mode for a visualization. See function help sections for details on display modes.

**Used By:**

PlotCapData, PlotCapGoodMeas

params.Nregs

**Description:**

This field determines which region labels of a mesh to display. The value must match those in the mesh.region field, and may either be a single number (“1”), or an array (“1:2”).

**Used By:**

PlotMeshSurface

params.cbmode

**Description:**

This field specifies whether to use custom colorbar axis labels.

**Used By:**

PlotSlices

params.cblabels

**Description:**

This field specifies custom colorbar axis labels. When cbmode==1, min defaults to 0 if PD==1, both default to +/- Scale if supplied. When cbmode==0, then cblabels dictates colorbar axis limits.

**Used By:**

PlotLRMeshes, PlotSlices

params.cbticks

**Description:**

When cbmode==1, this field specifies positions of tick marks on colorbar axis.

**Used By:**

PlotLRMeshes, PlotSlices

params.bins

**Description:**

This field determines the number of bins shown in histogram-type plots. The input must come in the form of either a single number (“50”), for which the bin limits will be automatically calculated from the data, or an array (“0:0.5:100”) that provides a set of bin limits.

**Used By:**

PlotHistogramData, PlotHistogramSTD

params.CH

**Description:**

This field determines whether to display crosshairs representing the current slices in a three-slice view. The value must be a logical 1 (for “on”) or 0 (for “off”).

**Used By:**

PlotSlices

params.slices

**Description:**

This field determines which slices are displayed in a 3D volume. Overrides interactive controls.

**Used By:**

PlotSlices

params.slices\_type

**Description:**

This field determines whether to use MATLAB indexing (‘idx’) or coordinates from the atlas space supplied in “infoAtlas” (‘coord’).

**Used By:**

PlotSlices

params.orientation

**Description:**

This field selects which orientation a 3D volume is displayed in. Possible values can be either ‘t’ for transverse, or ‘s’ for sagittal. The default is ‘t’.

**Used By:**

adjust\_brain\_pos, PlotMeshSurface, PlotSlices

params.view

**Description:**

This field sets the view perspective. Possible values are ‘lat’, ‘med’, ‘post’, and ‘dorsal’.

**Used By:**

adjust\_brain\_pos, PlotLRMeshes

params.ctx

**Description:**

This field defines inflation of a mesh. Possible values are ‘std’, ‘inf’, and ‘vinf’.

**Used By:**

adjust\_brain\_pos

params.alpha

**Description:**

This field defines the transparency of a mesh.

**Used By:**

PlotLRMeshes

params.OL

**Description:**

If “overlap” data is presented (OL==1), this field sets the interpolation method to “nearest”. Default is “linear”.

**Used By:**

vol2surf\_mesh

params.DR

**Description:**

This field selects the grayscale dynamic range of a visualization. The default is 1000.

**Used By:**

applycmap

params.Scale

**Description:**

This field selects the maximum value to which an image or volume is scaled. The default is (0.9 x the image max).

**Used By:**

applycmap, PlotInterpSurfMesh, PlotLRMeshes, PlotSlices

params.BG

**Description:**

This field determines the background color of a plot, as an RGB triplet. The default is gray, [0.5, 0.5, 0.5].

**Used By:**

applycmap, PlotCap, PlotMeshSurface

params.TC

**Description:**

This field sets the display to “true color” mapping.

**Used By:**

applycmap, PlotMeshSurface

params.PD

**Description:**

This field declares that the image is a positive definite set.

**Used By:**

applycmap, PlotLRMeshes, PlotSlices

params.Cmap

**Description:**

This field contains several subfields for the colormap:

* P – Positive domain colormap.
* N – Negative domain colormap.
* flipP – Inverts positive colormap.
* flipN – Inverts negative colormap.

**Used By:**

applycmap, PlotMeshSurface

params.Th

**Description:**

This field contains subfields that determine the threshold settings:

* P – Basemum threshold for displaying positive values.
* N – Maximum threshold for displaying negative values.

**Used By:**

applycmap, PlotInterpSurfMesh

params.Saturation

**Description:**

Field the size of data with values to set the coloring saturation. Must be within range [0, 1].

**Used By:**  
  
applycmap

params.kernel

**Description:**

A sampling kernel for time traces. Default value is a single point, other options are: ‘gaussian’ | ‘cube’ | ‘sphere’.

**Used By:**  
  
PlotSlicesTimeTrace

A

**Description:**

A MEAS x VOX matrix which maps the sensitivity of each source-detector pair to all of the voxels of the DOT space. Corresponds to (and thus named after) the “A” matrix in the traditional “inverse problem” equation commonly used in image reconstruction:

y = A \* x

E

**Description:**

A HB x WL sized matrix containing Beer-Lambert extinction coefficients of HbR and HbO at each WL.

dim

**Description:**

A structure describing the sensitivity matrix space.

* dim.xmin, dim.xmax, dim.ymin, dim.ymax, dim.zmin, dim.zmax
* dim.xv, dim.yv, dim.zv
* dim.nVx, dim.nVy, dim.nVz
* dim.nVt
* dim.sV
* dim.mmppix
* dim.center
* dim.Good\_Vox

dim.xmin, dim.xmax, dim.ymin, dim.ymax, dim.zmin, dim.zmax

**Description:**

Min and max coordinates in each dimension of the sensitivity matrix space.

dim.xv, dim.yv, dim.zv

**Description:**

Coordinates of each voxel along each dimension.

dim.nVx, dim.nVy, dim.nVz

**Description:**

Number of voxels in each dimension.

dim.nVt

**Description:**

Total number of voxels in the space.

dim.sV

**Description:**

Isometric voxel size, in millimeters.

dim.mmppix

**Description:**

Number of millimeters per voxel in each dimension (x, y, z). A negative value reflects that the direction is flipped in that dimension.

dim.center

**Description:**

A vector projecting from the center of voxel (0, 0, 0) to the center of the space described by dim.

dim.Good\_Vox

**Description:**

Voxels of the full cuboid volume described by dim that are populated with data in the associated data array.

header

**Description:**

A volumetric data header in 4dfp format.

* header.original\_header
* header.version\_of\_keys
* header.format
* header.conversion\_program
* header.filename
* header.bytes\_per\_pixel
* header.byte
* header.acq
* header.nDim
* header.nVx, header.nvY, header.nVz
* header.nVt
* header.mmx, header.mmy, header.mmz
* header.mmppix
* header.center

header.original\_header

**Description:**

If the header has been converted by NeuroDOT 2 to 4dfp from another file format, then the original data will always be kept.

header.version\_of\_keys

**Description:**

Key file version.

header.format

**Description:**

Format of raw data file (“float”, “double”, etc.).

header.conversion\_program

**Description:**

Program used to convert the space.

header.filename

**Description:**

Name of associated raw data file.

header.bytes\_per\_pixel

**Description:**

Number of bytes per pixel.

header.byte

**Description:**

Byte endian-ness.

header.acq

**Description:**

Canonical orientation of the raw data’s space.

header.nDim

**Description:**

Number of dimensions.

header.nVx, header.nVy, header.nVz

**Description:**

Number of voxels in each spatial dimension.

header.nVt

**Description:**

Number of time points.

header.mmx, header.mmy, header.mmz

**Description:**

Size of voxels in millimeters, in each dimension.

header.mmppix

**Description:**

Number of millimeters per voxel in each dimension (x, y, z). A negative value reflects that the direction is flipped in that dimension.

header.center

**Description:**

A vector projecting from the center of voxel (0, 0, 0) to the center of the space described by header.

mesh

**Description:**

A 3D surface mesh object with the following fields:

* mesh.nodes
* mesh.elements
* mesh.data
* mesh.region

mesh.nodes

**Description:**

A NODES x 3 column vector where each row represents the X, Y, and Z coordinates of a polygon vertex on the surface.

mesh.elements

**Description:**

An ELEMENTS x 3 column vector where each row represents the index of three NODES that create a polygon in 3D space.

mesh.data

**Description:**

An optional NODES x 1 array containing an interpolation of volumetric data at each node.

mesh.region

**Description:**

An optional NODEs x 1 array of region labels.

ops

**Description:**

A structure containing results from the Cap\_Fitter GUI, with the following fields:

* ops.rot\_x, ops.rot\_y, ops.rot\_z
* ops.trans\_x, ops.trans\_y, ops.trans\_z
* ops.flip\_x, ops.flip\_y, ops.flip\_z
* ops.scale\_m

ops.rot\_x, ops.rot\_y, ops.rot\_z

**Description:**

Rotation angles in each spatial dimension.

ops.trans\_x, ops.trans\_y, ops.trans\_z

**Description:**

Translation in each spatial dimension.

ops.flip\_x, ops.flip\_y, ops.flip\_z

**Description:**

Flip axes in each spatial dimension.

ops.scale\_m

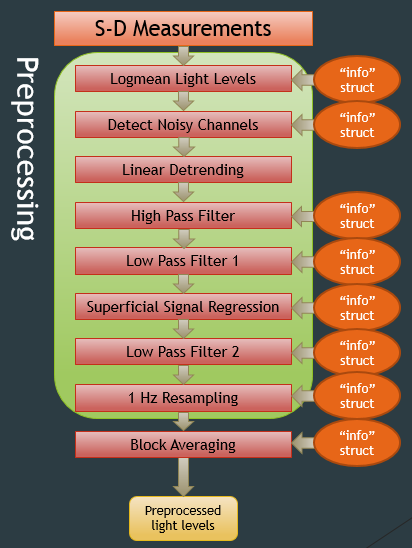
**Description:**

Scaling factor.

Pipelines

Preprocessing

**Diagram:**



**Description:**

This pipeline can be broken down into 9 main steps.

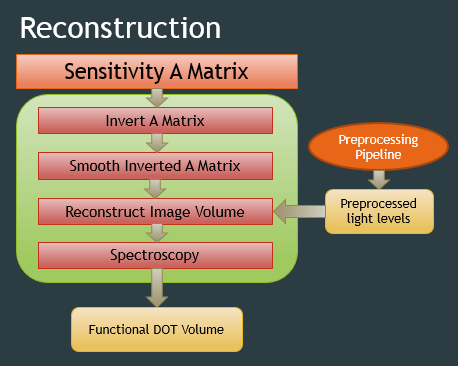
1. *Logmean Light Levels*. Because the mean light level variance between channels is much greater than the variance of light levels in each channel, we divide each channel by its mean light level and take the negative log. This essentially de-biases the detector differences from the signals.
2. *Detect Noisy Channels*. Some channels have too low SNR due to issues like poor coupling, and thus cannot be relied upon. These are filtered out with a straightforward threshold on noise, which is calculated as the standard deviation of each channel’s time trace.
3. *Linear Detrending*.
4. *High Pass Filter*. Removes long-term drift between stimulus blocks from the signal. Cutoff frequency is 0.02 Hz because stimulus blocks are at most 50-60 s long, and 1/50s = 0.02 Hz.
5. *Low Pass Filter 1*. This can technically be done at any point after the logmean, but we do it here. This filter has a cutoff frequency of 1 Hz, which is the framerate that the data is typically downsampled to in other pipelines (but was not included in this one). This filter removes the influence of any >1 Hz signals so they are not aliased by the downsampling.
6. *Superficial Signal Regression*. The blood vessels in the scalp contribute an interfering hemodynamic signal in the system. By taking the average of all channels for the first nearest neighbor (which are the most heavily influenced by hemodynamic signal) at each wavelength, we can regress this signal out of the data.
7. *Low Pass Filter 2*. The cutoff frequency for this filter is 0.5 Hz. It removes the signal due to the heart rate, which for most humans at rest is 60-70 beats per minute, and thus translates to a little less than 1 Hz.
8. *1 Hz Resampling*.
9. *Block Averaging*. This final step uses the experimental scheme to separate the time course of the data into equal-sized stimulus blocks. These are then averaged together and output as a result for analysis.

**Tutorial:**

“/NeuroDOT Base 2.2.0 Tutorial – DOT Processing Pipeline.pptx”

Reconstruction

**Diagram:**



**Description:**

This pipeline can be broken down into four main steps.

1. *Invert A Matrix.* The first reconstruction step is to invert the A matrix so as to satisfy the inverse problem. ND2 uses a Tikhonov inversion algorithm for this. It should be noted that the processing is done along two separate tracks - one for each wavelength.
2. *Smooth Inverted A Matrix.* The sensitivity matrix is smoothed with a Gaussian kernel.
3. *Reconstruct Image Volume.* The image is reconstructed from the raw data by solving the inverse problem.
4. *Spectroscopy.* Finally, the maps of light levels is multiplied by a set of extinction coefficients to get images of HbO and HbR concentrations.

**Tutorial:**

“/NeuroDOT Base 2.2.0 Tutorial – DOT Processing Pipeline.pptx”