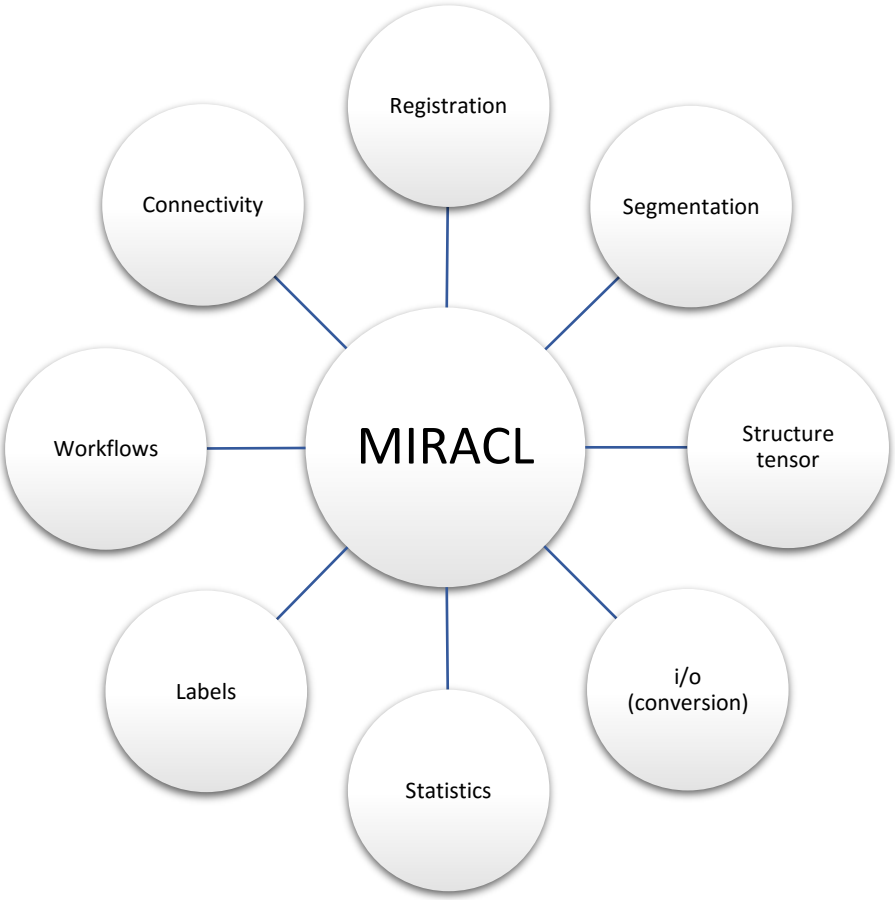
MIRACL DOCUMENTATION



- use bookmarks to navigate
- function specific documentation is included in the scripts and can be invoked by –h, -help, –-help
- we are adding more functions and documentation, if anything needs updating please let us know at mgoubran@stanford.edu

v. 1.0

CONNECTIVITY (CONNECT)

---

miracl_connect_label_graph_proj_dens.py

Get experiment project density & connectivity matrix for Allen label

Query Allen connectivity API for injection experiments & finds the experiment with highest proj volume
Outputs a connectivity graph of that experiment & its projection density images (as nii & tif)
If a label has no injection experiments, the connectivity atlas is searched for experiments for its parent label.

Search is performed for experiments on wildtype (C57BL/6J) mice

example: miracl_get_exp_conn_graph_proj_den.py -l

arguments (required):

l. Allen atlas label id

optional arguments:

-h, --help show this help message and exit

miracl_connect_ROI_matrix_connectogram.py

Get experiment connectivity matrix, projection & connectogram for ROI in Allen space

Usage: miracl_connect_ROI_matrix_connectogram.py -r [input Region Of Interest] -n [number of labels]

Finds the largest N Allen labels in the Region of Interest and extracts its N closely connected regions
(targets sorted by normalized projection volume) from the Allen Connectivity atlas.
Outputs a connectivity matrix of the primary injected sites (labels) & their most common targets.
Labels and targets are mutually exclusive (a primary injections is not chosen a target & vice versa).
If a label has no injection experiments, the connectivity atlas is searched for experiments for its parent label.
Querying from the Allen API requires an internet connection.

example: miracl_connect_ROI_matrix_connectogram.py -r my_roi_mask -n 15

arguments (required):

r. Input chosen region of interest (ROI) in Allen space

n. Output number of primary injection sites (also number of targer regions)
Limited by size of ROI (number of labels it contains)

optional arguments:

-h, --help show this help message and exit

miracl_connect_csd_tractography.sh

Performs CSD tractography and track density mapping for a seed region using mrtrix3

1) Computes response function of the diffusion data
2) Computes fiber orientation distribution (FOD) from the response function
3) Performs probabilistic CSD tractography using the iFOD2 method
4) Computes a tract density image (# of tracks / voxel)
5) Converts tracts to vtk format (for visualization in other software like 3DSlicer,Paraview..)

Usage: miracl_connect_csd_tractography.sh -d <DTI data> -m <binary mask> -r <bvecs> -b <bvals> -s <seed> -v <vox> -t <output tract name> -f <DTI dir> -e <exclusion seed> -p tckgen (tracking) options

Example: miracl_connect_csd_tractography.sh -d dti.nii.gz -m dti_mask.nii.gz -r dti.bvecs -b dti.bvals -s seed.nii.gz -v 1 -t CST -f dti_folder -e dti_exlcude_seed.nii.gz -p "-cutoff 0.2 -number 2000"

Required arguments:

d. DTI data

m. DTI mask

r. DTI bvecs

b. DTI bvals

s. Tractography seed

v. Tract density map voxel size

t. Output tract name

f. Directory containing DTI data

Optional arguments:

e. Exclusion seed for editing tracts

p. Tckgen (tractography) options

CONVERSION / GUI (IO)

---

miracl_io_convertTIFFtoNII.py

Converts tif files to nifti files while choosing a downsample factor, voxel spacing, channel info

A GUI will open to choose your:

- < Input clarity tif dir > : images of only one channel

For command-line / scripting

Usage: miracl_io_convertTIFFtoNII.py -f [Tiff folder] -o [out nii name]

Example: miracl_io_convertTIFFtoNII.py -f my_tifs -o stroke2

OR

example: miracl_io_convertTIFFtoNII.py -f my_tifs -d 3 -o stroke2 -cn 1 -cp C00 -ch Thy1YFP -vx 2.5 -vz 5

Arguments (required):

-f Input Clarity tif dir/folder

-o Output nii name (script will append downsample ratio and channel info to given name)

Optional arguments:

-d [Downsample ratio (default: 5)]

-cn [chan # for extracting single channel from multiple channel data (default: 1)]

-cp [chan prefix (string before channel number in file name). ex: C00]

-ch [output chan name (default: eyfp)]

-vx [original resolution in x-y plane in um (default: 5)]

-vz [original thickness (z-axis resolution / spacing between slices) in um (default: 5)]

-c [nii center (default: 5.7 -6.6 -4) corresponding to Allen atlas nii template]

-h, --help show this help message and exit

miracl_io_set_orient_gui.py

Uses user input (from a GUI) to generate an orientation code for orientation input data to "standard/Allen" orientation

Usage: miracl_io_set_orient_gui.py

A GUI will open to choose your:

- < Input CLARITY dir > : with tiff files (1 channel)

A GUI will open on the middle slice of the data. User can input slice number to navigate to a certain slice.

Then the user need to set the Orientation (Axial, Sagittal, Coronal) of the image from a drop-down menu.

They will also need to set the Annotation (Superior, Anterior, Left ...) of the Top, Right sides.

Finally, They need to set the annotation going into the page (located on the left of the data).

Then exit GUI.

The orientation code will be set based on the user entries and will be written to a text file (ort2std.txt) inside the dir.

miracl_io_convertDCMtoNII.sh

Converts dicoms in sub-directories to nii & renames sub-directories with sequence name

Usage: `basename $0` -p < parent dir >

arguments (required):

-p parent directory containing sub-directories with different sequences (with dcm files)

LABELS (LBLS)

---

miracl_lbls_generate_grand-parent_annotation.py

Generate multi-resolution atlases from Allen labels

Usage: miracl_lbls_generate_grand-parent_annotation.py -p [parent level (default: 3)] -m [hemisphere: split or combined (default: combined)] -v [voxel size in um: 10, 25 or 50 (default: 25)]

Generate multi-resolution atlases from Allen labels

example: miracl_lbls_generate_grand-parent_annotation.py -p 3 -m combined -v 10

required arguments:

-p PL, --pl PL parent level

optional arguments:

-h, --help show this help message and exit

-m HEMI, --hemi HEMI hemisphere mirrored or not

-v RES, --res RES voxel size in um

miracl_lbls_warp_to_clar_space.sh

Warps Allen annotations to the original high-res CLARITY space

Usage: miracl_lbls_warp_to_clar_space.sh

A GUI will open to choose your:

- < Input clarity registration folder >

- < Labels to warp>

Looks for ort2std.txt in directory where script is run

For command-line / scripting

Usage: `basename $0` -r [clarity registration dir] -l [labels in allen space to warp] -o [orient file]

Example: `basename $0` -r clar_reg_allen -l annotation_hemi_combined_25um_grand_parent_level_3.nii.gz -o ort2std.txt

arguments (required):

r. Input clarity registration dir

l. input Allen Labels to warp (in Allen space)

o. file with orientation to standard code

miracl_lbls_warp_to_mri_space.sh

Warps Allen annotations to the MRI space

Usage: miracl_lbls_warp_to_mri_space.sh

A GUI will open to choose your:

- < Input MRI registration folder >

- < Labels to warp>

For command-line / scripting

Usage: miracl_lbls_warp_to_mri_space.sh -i <input_down-sampled_MRI_nifti>

Example: miracl_lbls_warp_to_mri_space.sh -i Reference_channel_05_down.nii.gz

arguments (required):

r. Input MRI registration dir ("mri_allen_reg")

l. input Allen Labels to warp (in Allen space)

optional arguments:

miracl_lbls_generate_parents_at_depth.py

Generate parents labels at specific depth from Allen labels

usage:

```
    miracl_lbls_generate_parents_at_depth.py -d [depth] -m [hemisphere: split or combined (default:
combined)]  -v [voxel size in um: 10, 25 or 50 (default: 25)]

    Labels with higher depth are combined by their parent labels

    example: miracl_lbls_generate_parents_at_depth.py -d 6 -m combined -v 10
```

required arguments:

```
  -d DEPTH, --depth DEPTH
                        chosen depth
```

optional arguments:

```
  -h, --help            show this help message and exit

  -m HEMI, --hemi HEMI  hemisphere mirrored or not

  -v RES, --res RES     voxel size in um
```

miracl_lbls_stats.py

Computes Allen label stats of input volume

A GUI will open to choose your:

- < input volume >

- < registered Allen labels >

For command-line / scripting

Usage: miracl_lbls_stats.py -i [input volume] -l [reg Allen labels] -o [out csv]

Example: miracl_lbls_stats.py -i clarity_downsample_05x_virus_chan.nii.gz -l registered_labels.nii.gz -o label_stats.csv
-s Count

Arguments (required):

-i Input volume

-l Registered Allen labels

Optional arguments:

-o Output file name

-s Sort values by, options are:

Mean or StdD or Max or Min or Count or Vol(mm^3)

Mean -> mean intensity values

Count -> number of voxels

miracl_lbls_get_gp_volumes.py

Extracts volumes of labels of interest and their subdivisions from input label file (nii format)

usage:

miracl_lbls_get_gp_volumes.py -i [input labels image] -ln [label names] OR -la [label acronyms] -o [out csv]
-s sort by size [def: 0 -> no]

example: miracl_lbls_get_gp_volumes.py -i registered_labels.nii.gz -ln Thalamus Striatum -o gp_volumes.csv
OR
miracl_lbls_get_gp_volumes.py -i registered_labels.nii.gz -la TH STR -o gp_volumes.csv

required arguments:

-i IMG, --img IMG Input labels

-la ACRONYMS [ACRONYMS ...], --acronyms ACRONYMS [ACRONYMS ...]
Label acronyms

optional arguments:

-h, --help show this help message and exit

-ln NAMES [NAMES ...], --names NAMES [NAMES ...]
Label names

-s SORT, --sort SORT Sort by size

-o OUTFILE, --outfile OUTFILE

Output file

miracl_lbls_get_graph_info.py

Get label info from Allen atlas ontology graph

usage: miracl_lbls_get_graph_info.py -l [label id OR label name OR label acronym]

example: miracl_lbls_get_graph_info.py -l 672

OR

miracl_lbls_get_graph_info.py -l Caudoputamen

OR

miracl_lbls_get_graph_info.py -l CP

required arguments:

-l LBL, --lbl LBL input label

optional arguments:

-h, --help show this help message and exit

REGISTRATION (REG)

---

miracl_reg_clar-allen_whole_brain.sh

Register whole brain CLARITY to Allen

Registers CLARITY data (down-sampled images) to Allen Reference mouse brain atlas
Warps Allen annotations to the original high-res CLARITY space
Warps the higher-resolution CLARITY to Allen space

Usage: `basename $0` -i [input clarity nii] -o [orient code] -m [hemi mirror] -v [labels vox] -l [input labels] -s [side if hemisphere ony] -b [olfactory buld included]

Example: `basename $0` -i Reference_channel_05x_down.nii.gz -o ARS -m combined -v 25

arguments (required):

i. input down-sampled clarity nii
Preferably auto-fluorescence channel data (or Thy1_EYFP if no auto chan)

file name should have "##x_down" like "05x_down" (meaning 5x downsampled) -> ex. stroke13_05x_down_Ref_chan.nii.gz
[this should be accurate as it is used for allen label upsampling to clarity]

optional arguments:

o. orient code (default: ALS)
to orient nifti from original orientation to "standard/Allen" orientation

m. hemisphere mirror (default: combined)

warp allen labels with hemisphere split (Left different than Right labels) or combined (L & R same labels / Mirrored)
accepted inputs are: <split> or <combined>

v. labels voxel size/Resolution in um (default: 10)
accepted inputs are: 10, 25 or 50

l. input Allen labels to warp (default: annotation_hemi_combined_10um.nii.gz)
input labels could be at a different depth than default labels

If l. is specified (m & v cannot be specified)

s. side, if only registering a hemisphere instead of whole brain
accepted inputs are: rh (right hemisphere) or lh (left)

b. olfactory bulb included in brain, binary option (default: 0 -> not included)

Main Outputs

reg_final/clar_allen_space.nii.gz: Clarity data in Allen reference space

reg_final/clar_downsample_res(vox)um.nii.gz: Clarity data downsampled and oriented to "standard"

reg_final/annotation_hemi_(hemi)_(vox)um_clar_downsample.nii.gz: Allen labels registered to downsampled Clarity

reg_final/annotation_hemi_(hemi)_(vox)um_clar_vox.tif: Allen labels registered to oriented Clarity

reg_final/annotation_hemi_(hemi)_(vox)um_clar.tif: Allen labels registered to original (full-resolution) Clarity

- To visualize clarity data in Allen space - assuming chosen v/vox 10um
from command line:

itksnap -g $allen10 -o reg_final/clar_allen_space.nii.gz -s $lbls10 -l $snaplut

from GUI:

$allen10 = $MIRACL_HOME/atlases/ara/template/average_template_10um.nii.gz -> (Main Image)

$lbls10 = $MIRACL_HOME/atlases/ara/annotation/annotation_hemi_combined_10um.nii.gz ->
(Segmentation)

$snaplut = $MIRACL_HOME/atlases/ara/ara_snaplabels_lut.txt -> (Label Descriptions)

- To visualize Allen labels in downsampled clarity data space (from command line):

itksnap -g clar_downsample_res(vox)um.nii.gz -s
reg_final/annotation_hemi_(hemi)_(vox)um_clar_downsample.nii.gz

- Full resolution Allen labels in original clarity space (.tif) can be visualized by Fiji

miracl_reg_mri-allen.sh

Register MRI to Allen

Registers in-vivo or ex-vivo MRI data to Allen Reference mouse brain Atlas

Warps Allen annotations to the MRI space

Usage: `basename $0` -i [input invivo or exvivo MRI nii] -o [orient code] -m [hemi mirror] -v [labels vox] -l [input labels] -b [olfactory bulb] -s [skull strip] -n [no orient needed]

Example: `basename $0` -i inv_mri.nii.gz -o RSP -m combined -v 25

arguments (required):

i. input MRI nii
Preferably T2-weighted

optional arguments:

o. orient code (default: RSP)
to orient nifti from original orientation to "standard/Allen" orientation

m. hemisphere mirror (default: combined)
warp allen labels with hemisphere split (Left different than Right labels) or combined (L & R same labels / Mirrored)
accepted inputs are: <split> or <combined>

v. labels voxel size/Resolution in um (default: 10)
accepted inputs are: 10, 25 or 50

l. input Allen labels to warp (default: annotation_hemi_combined_10um.nii.gz)

input labels could be at a different depth than default labels

If l. is specified (m & v cannot be specified)

b. olfactory bulb included in brain, binary option (default: 0 -> not included)

s. skull strip or not, binary option (default: 1 -> skull-strip)

n. No orientation needed (input image in "standard" orientation), binary option (default: 0 -> orient)

miracl_reg_warp_clar_data_to_allen.sh

Warps downsampled CLARITY data/channels from native space to Allen atlas

Usage: miracl_reg_warp_clar_data_to_allen.sh

A GUI will open to choose your:

- < Input clarity registration folder >

- < downsampled CLARITY nii to warp >

- < ort2std.txt >

For command-line / scripting

Usage: miracl_reg_warp_clar_data_to_allen.sh -r [clarity registration dir] -i control03_05xdown_PIchan.nii.gz -o [orient file]

Example: miracl_reg_warp_clar_data_to_allen.sh -r clar_reg_allen -i control03_05xdown_PIchan.nii.gz -o ort2std.txt

arguments (required):

r. Input clarity registration dir

i. input downsampled CLARITY nii to warp

o. file with orientation to standard code

miracl_reg_warp_mr_data_to_allen.sh

Warps MRI data from native space to Allen atlas

Usage: miracl_reg_warp_mr_data_to_allen.sh

A GUI will open to choose your:

- < Input MRI registration folder >

- < MRI nii to warp >

- < ort2std.txt >

For command-line / scripting

Usage: miracl_reg_warp_mr_data_to_allen.sh -r [MRI registration dir] -i control03_05xdown_PIchan.nii.gz -o [orient file]

Example: miracl_reg_warp_mr_data_to_allen.sh -r mr_reg_allen -i inv_T2map.nii.gz -o ort2std.txt

arguments (required):

r. Input MRI registration dir

i. input MRI nii to warp

o. file with orientation to standard code

miracl_reg_check_results.py

checks registration results

usage: Usage: miracl_reg_check_results.py

A GUI will open to choose your:

- < reg final dir > : directory with final registration volumes and labels

---

For command-line / scripting

Usage: miracl_reg_check_results.py -f [reg final folder] -v [visualization software] -s [reg space (clarity or allen)]

Example: miracl_convertTifftoNii.py -f reg_final -v itk -s clarity

Arguments (required):

-f, --folder Input final registration folder

Optional arguments:

-v, --viz Visualization software: itkSNAP 'itk' (default) or freeview 'free'

-s, --space Registration Space of results: clarity (default) or allen

SEGMENTATION (SEG)

---

miracl_seg_clarity_neurons_wrapper.sh

Segments neurons in cleared mouse brain of sparse or nuclear stains in 3D

Usage: miracl_seg_clarity_neurons_wrapper.sh

A GUI will open to choose your input clarity folder (with .tif files)

The folder should contain data from one channel

For command-line / scripting

Usage: miracl_seg_clarity_neurons_wrapper.sh -d <clarity dir> -t <channel type: sparse or nuclear>

Example: miracl_seg_clarity_neurons_wrapper.sh -d my_clarity_tifs -t sparse -p Filter0001

arguments (required):

d. Input clarity directory (including .tif images of one channel) [folder name without spaces]

t. Channel type: sparse (like Thy1 YFP) or nuclear (like PI) (default = sparse)

Optional arguments:

p. Channel prefix & number if multiple channels (like Filter0001)

Main Outputs

segmentation/seg_sparse.tif (.mhd) or seg_nuclear.tif (.mhd) : segmentation image with all labels (cells)

segmentation/seg_bin_sparse.tif (.mhd) or seg_bin_nuclear.tif (.mhd) : binarized segmentation image

Results can be opened in Fiji for visualization

miracl_seg_voxelize_parallel.py

Voxelizes full-resolution segmentation results into density maps with Allen atlas resolution

Usage: mouse_seg_voxelize_parallel.py -s [segmentation tif]

example: mouse_seg_voxelize_parallel.py -s seg_sparse.tif

arguments (required):

s. Segmentation tif file

Main Outputs

voxelized_seg_sparse.(tif/nii) or nuclear (segmentation results voxelized to ARA resolution)
voxelized_seg_bin_sparse.(tif/nii) (binarized version)

miracl_seg_feat_extract.py

Computes segmentation features and summarizes them per label for input labels

Usage: mouse_feat_extract.py -s [segmentation tif] -l [Labels] -m [binary ROI mask]

Computes features of segmented image and summarizes them per label

example: mouse_feat_extract.py -s segmentation/voxelized_seg_sparse.tif -l reg_final/annotation_hemi_combined_25um_clar_vox.tif -m mask.tif

arguments (required):

s. Voxelized segmentation tif file

l. Allen labels (registered to clarity) used to summarize features

reg_final/annotation_hemi_(hemi)_(vox)um_clar_vox.tif

optional arguments:

m. Mask to choose a region of interest (ROI) to analyze (binary image, 0 value outside ROI)

Will extract features from allen labels within ROI
Should be in clarity space

Main Outputs

clarity_segmentation_features_ara_labels.csv (segmentation features summarized per ARA lables)

STATISTICS (STATS)

---

miracl_stats_paired_ttest_ipsi_contra.py

Computes paired t-test test between both hemispheres for all labels across mice

usage: Usage: miracl_stats_paired_ttest_ipsi_contra.py -d [folder with feature extraction csv files]

Looks for feature extraction csv files within input directory

Outputs csv,xlsx files with stats results & a nifti image with label values corresponding to p-values of the t-test

example: miracl_stats_paired_ttest_ipsi_contra.py -d feat_extract_csv

required arguments:

-d, --dir dir with csv files

optional arguments:

-h, --help show this help message and exit

miracl_stats_correlate_parms_w_feats.py

Computes correlations between MRI parameters & CLARITY features for all labels across mice

usage: Usage: miracl_stats_correlate_parms_w_feats.py -d [folder with feature extraction csv files]

Looks for parms & feature extraction csv files within input directory

Outputs csv,xlsx files with stats results & a nifti image with label values corresponding to Rs-value of the Spearman Correlation

example: miracl_stats_correlate_parms_w_feats.py -d parms_feat_extract_csv

required arguments:

-d, --dir dir with csv files

optional arguments:

-h, --help show this help message and exit

STRUCTURE TENSOR ANALYSIS (STA)

---

miracl_sta_track_primary_eigen.sh

Performs Structure Tensor Analysis (STA)

Runs Matlab scripts

Usage: miracl_sta_track_primary_eigen.sh

A GUI will open to choose your input down-sampled clarity nifti, brain mask, seed mask

---

For command-line / scripting

Usage: miracl_sta_track_primary_eigen.sh -i <down-sampled clarity nifti> -g <dog sigma> -k <gaussian sigma> -a <tracking angle threshold>
-b <brain mask> -s <seed> -o <output dir>

Example: miracl_sta_track_primary_eigen.sh -i clarity_03x_down_virus_chan.nii.gz -g 0.5 -k 0.5 -a 25 -b brain_mask.nii.gz -s seed.nii.gz -o sta

required arguments:

f. Input down-sampled clarity nifti (.nii/.nii.gz)

g. Derivative of Gaussian (dog) sigma

k. Gaussian smoothing sigma

a. Tracking angle threshold

b. Brain mask (.nii/.nii.gz)

s. Seed mask (.nii/.nii.gz)

optional arguments:

o. Out dir

Main Outputs

fiber.trk => Fiber tracts

WORKFLOWS (FLOW)

---

miracl_workflow_registration_clarity-allen_wb.sh

Workflow (wrapper) combining multiple MIRACL registration/io functions

Sets orientation of input data using a GUI
Converts TIFF to NII
Registers CLARITY data (down-sampled images) to Allen Reference mouse brain atlas
Warps Allen annotations to the original high-res CLARITY space
Warps the higher-resolution CLARITY to Allen space

Executes:

io/miracl_io_set_orient_gui.py (if run in GUI mode)
io/miracl_io_convertTIFFtoNII.py
reg/miracl_reg_clar-allen_whole_brain.sh

Usage: `basename $0`

A GUI will open to set data orientation
For the "miracl_io_convertTIFFtoNII.py" & "miracl_reg_clar-allen_whole_brain.sh" default parameters will be chosen

---

For command-line / scripting

Usage: `basename $0` -f [Tiff folder]

Example: `basename $0` -f my_tifs -n "-d 5" -r "-o ARS -m combined -v 25"

arguments (required):

f. Input Clarity tif folder/dir

optional arguments (don't forget the quotes):

conversion to nii (invoked by -n " "):

d. [Downsample ratio (default: 5)]
cn. [chan # for extracting single channel from multiple channel data (default: 1)]
cp. [chan prefix (string before channel number in file name). ex: C00]
ch. [output chan name (default: eyfp)]
vx. [original resolution in x-y plane in um (default: 5)]
vz. [original thickness (z-axis resolution / spacing between slices) in um (default: 5)]
c. [nii center (default: 5.7 -6.6 -4) corresponding to Allen atlas nii template]

Registration (invoked by -r " "):

o. Orient code (default: ALS)
to orient nifti from original orientation to "standard/Allen" orientation

m. Warp allen labels with hemisphere split (Left different than Right labels) or combined (L & R same labels / Mirrored)
accepted inputs are: <split> or <combined> (default: combined)

v. Labels voxel size/Resolution of labels in um
accepted inputs are: 10, 25 or 50 (default: 10)

l. image of input Allen Labels to warp (default: annotation_hemi_split_10um.nii.gz - which are at a resolution of 0.01mm/10um)

input could be at a different depth than default labels

If l. is specified (m & v cannot be specified)

s. side, if only registering a hemisphere instead of whole brain
accepted inputs are: rh (right hemisphere) or lh (left)

b. olfactory bulb included in brain, binary option (default: 0 -> not included)

Main Outputs

reg_final/clar_allen_space.nii.gz: Clarity data in Allen reference space

reg_final/clar_downsample_res(vox)um.nii.gz : Clarity data downsampled and oriented to "standard"

reg_final/annotation_hemi_(hemi)_(vox)um_clar_downsample.nii.gz : Allen labels registered to downsampled Clarity

reg_final/annotation_hemi_(hemi)_(vox)um_clar.tif: Allen labels registered to original Clarity

- To visualize clarity data in Allen space - assuming chosen v/vox 10um
from command line:

itksnap -g $allen10 -o reg_final/clar_allen_space.nii.gz -s $lbls10 -l $snaplut

from GUI:

$allen10 = $MIRACL_HOME/atlases/ara/template/average_template_10um.nii.gz -> (Main Image)

$lbls10 = $MIRACL_HOME/atlases/ara/annotation/annotation_hemi_combined_10um.nii.gz -> (Segmentation)

$snaplut = $MIRACL_HOME/atlases/ara/ara_snaplabels_lut.txt -> (Label Descriptions)

- To visualize Allen labels in downsampled clarity data space (from command line):

itksnap -g clar_downsample_res(vox)um.nii.gz -s reg_final/annotation_hemi_(hemi)_(vox)um_clar_downsample.nii.gz

- Full resolution Allen labels in original clarity space (.tif) can be visualized by Fiji

miracl_workflow_segmentation_clarity.sh

Workflow (wrapper) combining multiple MIRACL segmentation functions

Segments neurons in cleared mouse brain of sparse or nuclear stains in 3D
Voxelizes segmentation results into density maps with Allen atlas resolution
Computes features of segmented image and summarizes them per label

Executes:

seg/miracl_seg_clarity_neurons_wrapper.sh
seg/miracl_seg_voxelize_parallel.py
seg/miracl_seg_feat_extract.py

Usage: `basename $0`

A GUI will open to choose folder with tif files for segmentation (folder must have one channel)

Channel type is assumed to be sparse

For feature extraction the "reg_final/annotation_hemi_combined_(vox)um_clar_vox.tif" file must exist where the script is started

---

For command-line / scripting

Usage: `basename $0` -f [Tiff folder]

Example: `basename $0` -f my_tifs -t nuclear -s "-p C000" -e "-l reg_final/annotation_hemi_combined_25um_clar_vox.tif"

arguments (required):

f. Input Clarity tif folder/dir [folder name without spaces]

t. Channel type: sparse (like Thy1 YFP) or nuclear (like PI)

optional arguments (don't forget the quotes):

Segmentation (invoked by -s " "):

p. Channel prefix & number if multiple channels (like Filter0001)

Feature extraction (invoked by -e " "):

l. Allen labels (registered to clarity) used to summarize features

reg_final/annotation_hemi_(hemi)_(vox)um_clar_vox.tif

Main Outputs

segmentation/seg.tif (.mhd) or seg_nuclear.tif (.mhd) : segmentation image with all labels (cells)
segmentation/seg_bin.tif (.mhd) or seg_bin_nuclear.tif (.mhd) : binarized segmentation image

voxelized_seg.(tif/nii) (segmentation results voxelized to ARA resolution)
voxelized_seg_bin.(tif/nii) (binarized version)

clarity_segmentation_features_ara_labels.csv (segmentation features summarized per ARA labels)

Results can be opened in Fiji for visualization

miracl_workflow_sta.sh

Workflow (wrapper) for structure tensor analysis (STA):

1) Converts Tiff stack to nii (& down-sample)
2) Uses registered labels to create seed mask & creates brain mask
3) Run STA analysis

Executes:

io/miracl_io_convertTifftoNII.py
utils/miracl_extract_lbl.py
utils/miracl_create_brainmask.py
sta/miracl_sta_track_primary_eigen.py

Usage: miracl_workflow_sta.sh

A GUI will open to choose folder with tif files for STA and the registered Allen labels

and choosing STA parameters

For command-line / scripting

Usage: miracl_workflow_sta.sh -f [Tiff folder] -o [output nifti] -l [Allen seed label] -r [Reg final dir] -d [downsample ratio]

Example: miracl_workflow_sta.sh -f my_tifs -o clarity_virus_05xdown.nii.gz -l PL -r clar_reg_final -n "-d 5 -ch AAV" -t "-g 0.5 -k 0.5 -a 25"

arguments (required):

f. Input Clarity tif folder/dir [folder name without spaces]

o. Output nifti

l. Seed label abbreviation (from Allen atlas ontology)

r. CLARITY final registration folder

optional arguments (do not forget the quotes):

d. Downsample ratio (default: 5)

conversion (invoked by -n " "):

cn. [chan # for extracting single channel from multiple channel data (default: 1)]
cp. [chan prefix (string before channel number in file name). ex: C00]
ch. [output chan name (default: AAV)]
vx. [original resolution in x-y plane in um (default: 5)]
vz. [original thickness (z-axis resolution / spacing between slices) in um (default: 5)]
c. [nii center (default: 5.7 -6.6 -4) corresponding to Allen atlas nii template]

sta parameters (invoked by -t " "):

g. [Derivative of Gaussion (dog) sigma]
k. [Gaussian smoothing sigma]
a. [Tracking angle threshold]

miracl_workflow_multiple_mice.sh

Wrapper for running a MIRACL script on multiple mice

Script has to run in command-line mode

Usage: miracl_workflow_multiple_mice.sh -sm [script module] -sc [script] -id [input directory] -opt [options] -nj [#_of_jobs] -tj [time_between_batches(min)]

Example: miracl_workflow_multiple_mice.sh -st reg -sc miracl_clar-allen_reg_ants.sh -id my_data_dir -opt "-m combined" -nj 3 -tj 10

arguments (required):

st. script module / type
example: reg or seg or lbls or connect

sc. script name
example: miracl_clar-allen_reg_ants.sh

id. input directory with multiple mice data

opt. input options for script to run

nj. number of mice to run simultaneously - in parallel "||"

tj. time for computer to "sleep" between parallel jobs